



TÉCNICAS DE ANÁLISIS DE DATOS

APLICACIONES PRÁCTICAS UTILIZANDO MICROSOFT EXCEL Y WEKA

Jesús García Herrero
José Manuel Molina López

2012

PRÓLOGO

Estos apuntes pretenden dar una visión general de las técnicas de análisis de datos y de las aplicaciones que las implementan, permitiendo entender los conceptos y algoritmos sobre los que se basan las técnicas así como el resultado de su aplicación sobre diversas fuentes de ficheros.

Estos apuntes son una recolección de información de muy variadas fuentes, páginas de intenet, artículos etc.. todas ellas aparecen citadas. De entre todas ellas cabe resaltar el trabajo fin de carrera de David Sánchez titulado “Data Mining mediante Sistemas Clasificadores Genéticos. Análisis comparativo con las técnicas clásicas implementadas en WEKA”, en la titulación de Ingeniería Informática (Julio 2003) donde se realiza un gran esfuerzo por explicar el funcionamiento interno de la herramienta WEKA y de dónde se ha extraído la información acerca de las clases y el código que implementa los algoritmos para estos apuntes. Así también resulta necesario resaltar la tesis doctoral de Félix Chamorro, ya que el capítulo 2 (el estado del arte) se pormenorizan todas las técnicas de análisis de datos y que ha sido utilizado para la elaboración de estos apuntes.

Esperamos que estos apuntes sean de utilidad para los alumnos que se acerquen al análisis de datos y en particular para aquellos que tengan interés en aplicar los conocimientos teóricos en el campo de la práctica.

José Manuel Molina López — Jesús García Herrero

Índice

CAPÍTULO 1. INTRODUCCIÓN 1

1.1. KDD Y MINERÍA DE DATOS 1

1.1.2. EL PROCESO DE KDD 3

1.1.3. MINERÍA DE DATOS 5

1.1.4. TECNOLOGÍAS DE APOYO 6

1.1.5. ÁREAS DE APLICACIÓN 9

1.1.6. TENDENCIAS DE LA MINERÍA DE DATOS 13

1.2. MINERÍA DE DATOS Y ALMACENAMIENTO DE DATOS 14

1.2.1. ARQUITECTURA, MODELADO, DISEÑO, Y ASPECTOS DE LA ADMINISTRACIÓN 14

1.2.2. DATA MINING Y FUNCIONES DE BASES DE DATOS 16

1.2.3. DATA WAREHOUSE 17

1.2.4. DATA WAREHOUSE Y DATA MINING 21

1.3. HERRAMIENTAS COMERCIALES DE ANÁLISIS DE DATOS 22

1.4. ARQUITECTURA SOFTWARE PARA DATA MINING 33

1.4.2. ARQUITECTURA FUNCIONAL 35

1.4.3. ARQUITECTURA DEL SISTEMA 36

1.4.4. EL DATA MINING EN LA ARQUITECTURA DEL SISTEMA 38

CAPÍTULO 2. ANÁLISIS ESTADÍSTICO MEDIANTE EXCEL 41

2.1. ANÁLISIS DE UNA VARIABLE. ESTADÍSTICA DESCRIPTIVA E INFERENCIA 43

2.2. TÉCNICAS DE EVALUACIÓN DE HIPÓTESIS 57

2.2.1. ANÁLISIS DE RELACIONES ENTRE ATRIBUTOS 57

2.2.2. RELACIÓN ENTRE VARIABLES NOMINALES-NOMINALES 57

2.2.3. RELACIONES NUMÉRICAS-NOMINALES 59
2.2.3.1. Comparación de dos medias 59
2.2.3.2. Análisis de la varianza 61

2.2.4. RELACIONES NUMÉRICAS-NUMÉRICAS: 64
2.2.4.1. Regresión lineal 64

2.2.5. EVALUACIÓN DEL MODELO DE REGRESIÓN 65
2.2.5.1. Medidas de Calidad 65
2.2.5.2. Test de Hipótesis sobre modelo de regresión 66

2.3. EJEMPLOS DE APLICACIÓN DE TÉCNICAS DE EVALUACIÓN DE HIPÓTESIS 67

2.3.1. EJEMPLOS DE VALIDACIÓN DE HIPÓTESIS 67

2.4. TÉCNICAS CLÁSICAS DE CLASIFICACIÓN Y PREDICCIÓN 76

2.4.1. CLASIFICACIÓN BAYESIANA: 80

2.4.2. REGRESIÓN LINEAL 90

CAPÍTULO 3. TÉCNICAS DE MINERÍA DE DATOS BASADAS EN APRENDIZAJE AUTOMÁTICO 96

3.1. TÉCNICAS DE MINERÍA DE DATOS 96

3.2. CLUSTERING. (“SEGMENTACIÓN”) 98

3.2.1. CLUSTERING NUMÉRICO (K-MEDIAS) 99

3.2.2. CLUSTERING CONCEPTUAL (COBWEB) 100

3.2.3. CLUSTERING PROBABILÍSTICO (EM) 104

3.3. REGLAS DE ASOCIACIÓN 107

3.4. LA PREDICCIÓN 110

3.4.1. REGRESIÓN NO LINEAL. 110

3.4.2. ÁRBOLES DE PREDICCIÓN 111

3.4.3. ESTIMADOR DE NÚCLEOS 115

3.5. LA CLASIFICACIÓN 120

3.5.1. TABLA DE DECISIÓN 121

3.5.2. ÁRBOLES DE DECISIÓN 123

3.5.3. REGLAS DE CLASIFICACIÓN 135

3.5.4. CLASIFICACIÓN BAYESIANA 140
3.5.5. APRENDIZAJE BASADO EN EJEMPLARES 145
3.5.6. REDES DE NEURONAS 153
3.5.7. LÓGICA BORROSA (“FUZZY LOGIC”) 157
3.5.8. TÉCNICAS GENÉTICAS: ALGORITMOS GENÉTICOS (“GENETIC ALGORITHMS”) 157

CAPÍTULO 4. TÉCNICAS DE ANÁLISIS DE DATOS EN WEKA 159

INTRODUCCIÓN 159

PREPARACIÓN DE LOS DATOS 160

MUESTRA DE DATOS 160
OBJETIVOS DEL ANÁLISIS 161

EJECUCIÓN DE WEKA 162

PREPROCESADO DE LOS DATOS 164

CARACTERÍSTICAS DE LOS ATRIBUTOS 165
TRABAJO CON FILTROS. PREPARACIÓN DE FICHEROS DE MUESTRA 167
Filtros de atributos 168
Filtros de instancias 172

VISUALIZACIÓN 173

REPRESENTACIÓN 2D DE LOS DATOS 173
FILTRADO “GRÁFICO” DE LOS DATOS 177

ASOCIACIÓN 178

AGRUPAMIENTO 183

AGRUPAMIENTO NUMÉRICO 184
AGRUPAMIENTO SIMBÓLICO 189

CLASIFICACIÓN 191

MODOS DE EVALUACIÓN DEL CLASIFICADOR 192
SELECCIÓN Y CONFIGURACIÓN DE CLASIFICADORES 195
PREDICCIÓN NUMÉRICA 203
APRENDIZAJE DEL MODELO Y APLICACIÓN A NUEVOS DATOS. 209

SELECCIÓN DE ATRIBUTOS 211

CAPÍTULO 5. IMPLEMENTACIÓN DE LAS TÉCNICAS DE ANÁLISIS DE DATOS EN WEKA 215

5.1. UTILIZACIÓN DE LAS CLASES DE WEKA EN PROGRAMAS INDEPENDIENTES 215

5.2. TABLA DE DECISIÓN EN WEKA 215

5.3. ID3 EN WEKA 216

5.4. C4.5 EN WEKA (J48) 216

5.5. ÁRBOL DE DECISIÓN DE UN SOLO NIVEL EN WEKA 219

5.6. 1R EN WEKA 220

5.7. PRISM EN WEKA 221

5.8. PART EN WEKA 221

5.9. *NAIVE* BAYESIANO EN WEKA 222

5.10. VFI EN WEKA 223

5.11. KNN EN WEKA (IBK) 224

5.12. K* EN WEKA 226

5.13. REDES DE NEURONAS EN WEKA 227

5.14. REGRESIÓN LINEAL EN WEKA 228

5.15. REGRESIÓN LINEAL PONDERADA LOCALMENTE EN WEKA 230

5.16. M5 EN WEKA 231

5.17. KERNEL DENSITY EN WEKA 232

5.18. *K*-MEANS EN WEKA 234

5.19. COBWEB EN WEKA 234

5.20. EM EN WEKA 235

5.21. ASOCIACIÓN A PRIORI EN WEKA 236

CAPÍTULO 6. EJEMPLOS SOBRE CASOS DE ESTUDIO 239

BIBLIOGRAFÍA 240

Capítulo 1. Introducción

En este texto se estudia uno de los campos que más se están estudiando en estos días: La extracción de conocimiento a partir de fuentes masivas de datos. Para ello se emplean las denominadas técnicas de minería de datos, que son algoritmos capaces de obtener relaciones entre distintos atributos o conceptos para ayudar, por ejemplo, a la toma de decisiones.

Además de las técnicas estadísticas se estudian las técnicas de Minería de Datos [Data Mining] basadas en técnicas de aprendizaje automático que se implementan en una herramienta de minería de datos de libre distribución: WEKA. Esta herramienta permite, a partir de ficheros de texto en un formato determinado, utilizar distintos tipos de técnicas para extraer información.

A continuación se definen los conceptos fundamentales empleados en el texto: KDD y, sobretodo, minería de datos, así como sus principales características. Posteriormente se comenta la estructura del proyecto.

1.1. KDD y Minería de Datos

Hoy en día, la cantidad de datos que ha sido almacenada en las bases de datos excede nuestra habilidad para reducir y analizar los datos sin el uso de técnicas de análisis automatizadas. Muchas bases de datos comerciales transaccionales y científicas crecen a una proporción fenomenal.

KDD [Knowledge Discovery in Databases] [PSF91] es el proceso completo de extracción de información, que se encarga además de la preparación de los datos y de la interpretación de los resultados obtenidos. KDD se ha definido como "el proceso no trivial de identificación en los datos de patrones válidos, nuevos, potencialmente útiles, y finalmente comprensibles" [FAYY96]. Se trata de interpretar grandes cantidades de datos y encontrar relaciones o patrones. Para conseguirlo harán falta técnicas de aprendizaje automático [Machine Learning] [MBK98], estadística [MIT97, DEGR86], bases de datos [CODD70], técnicas de representación del conocimiento, razonamiento basado en casos [CBR, Case Based Reasoning], razonamiento aproximado, adquisición de conocimiento, redes de neuronas y visualización de datos. Tareas comunes en KDD son la inducción de reglas, los problemas de clasificación y clustering, el reconocimiento de patrones, el modelado predictivo, la detección de dependencias, etc.

KDD es un campo creciente: hay muchas metodologías del descubrimiento del conocimiento en uso y bajo desarrollo. Algunas de estas técnicas son genéricas, mientras otros son de dominio específico.

Los datos recogen un conjunto de hechos (una base de datos) y los patrones son expresiones que describen un subconjunto de los datos (un modelo aplicable a ese subconjunto). KDD involucra un proceso iterativo e interactivo de búsqueda de modelos, patrones o parámetros. Los patrones descubiertos han de ser válidos, novedosos para el sistema (para el usuario siempre que sea posible) y potencialmente útiles.

Se han de definir medidas cuantitativas para los patrones obtenidos (precisión, utilidad, beneficio obtenido...). Se debe establecer alguna medida de interés [interestingness] que considere la validez, utilidad y simplicidad de los patrones obtenidos mediante alguna de las técnicas de Minería de Datos. El objetivo final de todo esto es incorporar el conocimiento obtenido en algún sistema real, tomar decisiones a partir de los resultados alcanzados o, simplemente, registrar la información conseguida y suministrársela a quien esté interesado.

Ha llegado un momento en el que disponemos de tanta información que nos vemos incapaces de sacarle provecho. Los datos tal cual se almacenan [raw data] no suelen proporcionar beneficios directos. Su valor real reside en la información que podamos extraer de ellos: información que nos ayude a tomar decisiones o a mejorar nuestra comprensión de los fenómenos que nos rodean.

Se requiere de grandes cantidades de datos que proporcionen información suficiente para derivar un conocimiento adicional. Dado que se requieren grandes cantidades de datos, es esencial el proceso de la eficiencia. La exactitud es requerida para asegurar que el descubrimiento del conocimiento es válido. Los resultados deberán ser presentados de una manera entendible para el ser humano. Una de las premisas mayores de KDD es que el conocimiento es descubierto usando técnicas de aprendizaje inteligente que van examinando los datos a través de procesos automatizados. Para que una técnica sea considerada útil para el descubrimiento del conocimiento, éste debe ser interesante; es decir, debe tener un valor potencial para el usuario.

KDD proporciona la capacidad para descubrir información nueva y significativa usando los datos existentes. KDD se define como: *"The nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data"* en Fayyad, Piatetsky-Shapiro & Smyth: "From data mining to knowledge discovery: An overview" Advances in Knowledge Discovery and Data Mining (AAAI / MIT Press, 1996) y se puede resumir en la Figura 1.

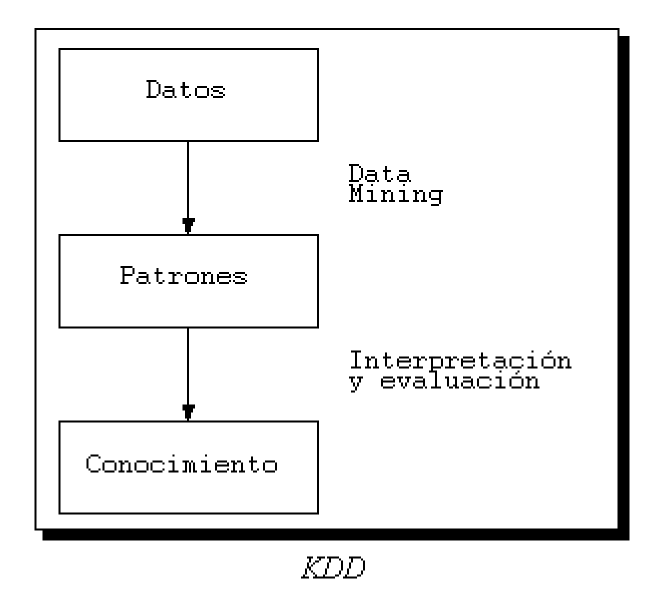


Figura 1.1: *Esquema del proceso de KDD*

KDD rápidamente excede la capacidad humana para analizar grandes cantidades de datos. La cantidad de datos que requieren procesamiento y análisis en grandes bases de datos exceden las capacidades humanas y la dificultad de transformar los datos con precisión es un conocimiento que va más allá de los límites de las bases de datos tradicionales. Por consiguiente, la utilización plena de los datos almacenados depende del uso de técnicas del descubrimiento del conocimiento.

La utilidad de aplicaciones futuras en KDD es de largo alcance. KDD puede usarse como un medio de recuperación de información, de la misma manera que los agentes inteligentes realizan la recuperación de información en el Web. Nuevos modelos o tendencias en los datos podrán descubrirse usando estas técnicas. KDD también puede usarse como una base para las interfaces inteligentes del mañana, agregando un componente del descubrimiento del conocimiento a un sistema de bases de datos o integrando KDD con las hojas de cálculo y visualizaciones.

1.1.2. El proceso de KDD

El proceso de KDD se inicia con la identificación de los datos. Para ello hay que imaginar qué datos se necesitan, dónde se pueden encontrar y cómo conseguirlos. Una vez que se dispone de datos, se deben seleccionar aquellos que sean útiles para los objetivos propuestos. Se preparan, poniéndolos en un formato adecuado.

Una vez se tienen los datos adecuados se procede a la minería de datos, proceso en el que se seleccionarán las herramientas y técnicas adecuadas para lograr los objetivos pretendidos. Y tras este proceso llega el análisis de resultados, con lo que se obtiene el conocimiento pretendido.

En la figura 1.2 se muestra la metodología que debe seguirse para obtener conocimiento a partir de los datos que se encuentran en la base de datos.

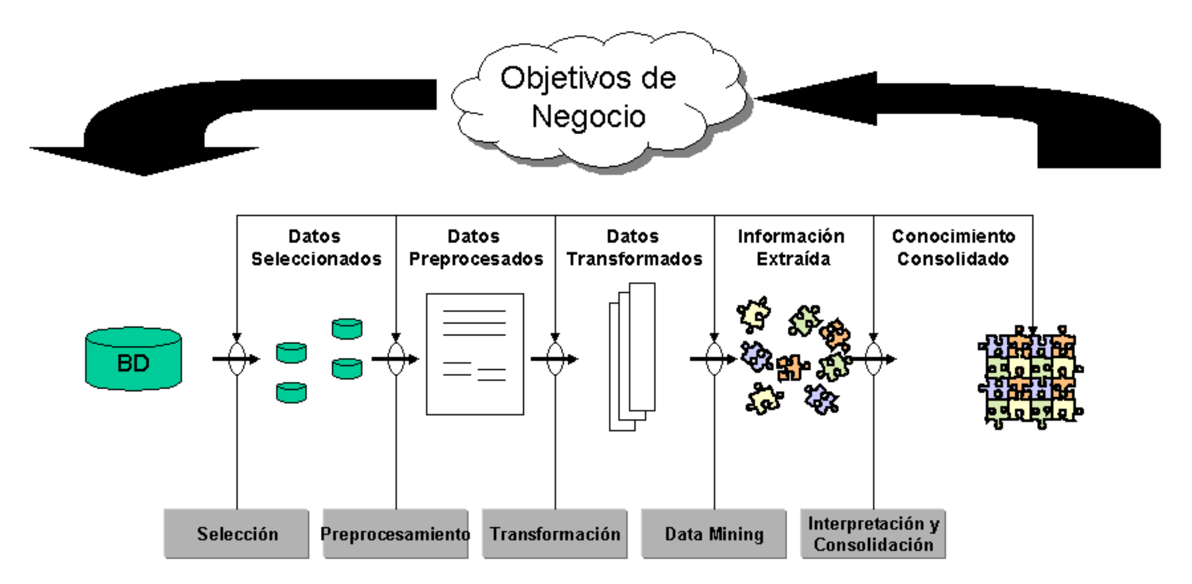


Figura 1.2: *Metodología para el descubrimiento de conocimiento en bases de datos.*

KDD es un proceso interactivo e iterativo, que involucra numerosos pasos e incluye muchas decisiones que deben ser tomadas por el usuario, y se estructura en las siguientes etapas [FAYY96]:

- Comprensión del dominio de la aplicación, del conocimiento relevante y de los objetivos del usuario final.
- Creación del conjunto de datos: consiste en la selección del conjunto de datos, o del subconjunto de variables o muestra de datos, sobre los cuales se va a realizar el descubrimiento.
- Limpieza y preprocesamiento de los datos: Se compone de las operaciones, tales como: recolección de la información necesaria sobre la cual se va a realizar el proceso, decidir las estrategias sobre la forma en que se van a manejar los campos de los datos no disponibles, estimación del tiempo de la información y sus posibles cambios.
- Reducción de los datos y proyección: Encontrar las características más significativas para representar los datos, dependiendo del objetivo del proceso. En este paso se pueden utilizar métodos de transformación para reducir el número efectivo de variables a ser consideradas o para encontrar otras representaciones de los datos.
- Elegir la tarea de Minería de Datos: Decidir si el objetivo del proceso de KDD es: Regresión, Clasificación, Agrupamiento, etc.
- Elección del algoritmo(s) de Minería de Datos: Selección del método(s) a ser utilizado para buscar los patrones en los datos. Incluye además la decisión sobre que modelos y parámetros pueden ser los más apropiados.
- Minería de Datos: Consiste en la búsqueda de los patrones de interés en una determinada forma de representación o sobre un conjunto de

representaciones, utilizando para ello métodos de clasificación, reglas o árboles, regresión, agrupación, etc.

- Interpretación de los patrones encontrados. Dependiendo de los resultados, a veces se hace necesario regresar a uno de los pasos anteriores.

- Consolidación del conocimiento descubierto: consiste en la incorporación de este conocimiento al funcionamiento del sistema, o simplemente documentación e información a las partes interesadas.

El proceso de KDD puede involucrar varias iteraciones y puede contener ciclos entre dos de cualquiera de los pasos. La mayoría de los trabajos que se han realizado sobre KDD se centran en la etapa de minería. Sin embargo, los otros pasos se consideran importantes para el éxito del KDD. Por eso aunque la Minería de Datos es una parte del proceso completo de KDD [FAYY96], en buena parte de la literatura los términos Minería de Datos y KDD se identifican como si fueran lo mismo.

En la figura 1.3 se muestra el esfuerzo que requiere cada fase del proceso de KDD.

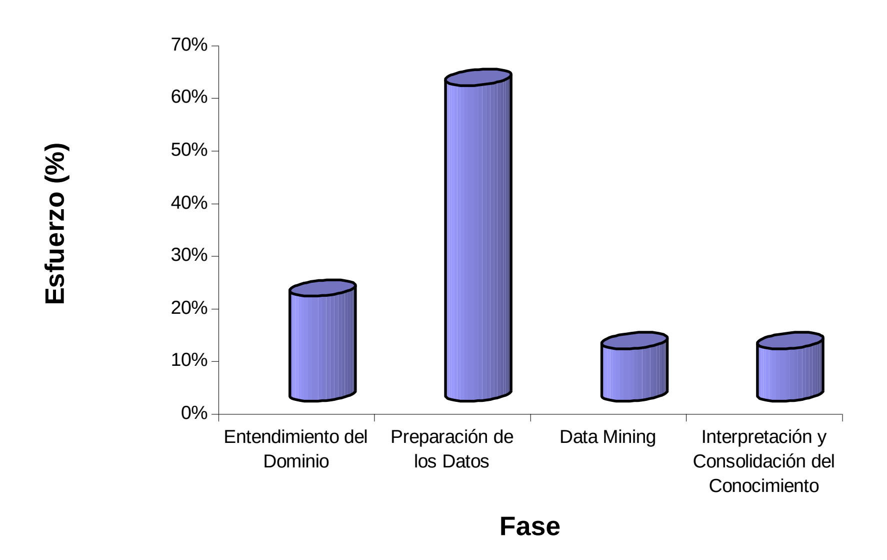


Figura 1.3: *Esfuerzo requerido por cada fase del proceso de KDD.*

Como se observa en la figura 1.3, gran parte del esfuerzo del proceso de KDD recae sobre la fase de preparación de los datos, fase crucial para tener éxito como ya se comentó anteriormente.

1.1.3. Minería de Datos

Minería de Datos es un término genérico que engloba resultados de investigación, técnicas y herramientas usadas para extraer información útil de grandes bases de datos. Si bien Minería de Datos es una parte del proceso completo de KDD, en buena parte de la literatura los términos Minería de Datos y KDD se identifican como si fueran lo mismo. Concretamente, el término

Minería de Datos es usado comúnmente por los estadísticos, analistas de datos, y por la comunidad de administradores de sistemas informáticos como todo el proceso del descubrimiento, mientras que el término KDD es utilizado más por los especialistas en Inteligencia Artificial.

El análisis de la información recopilada (por ejemplo, en un experimento científico) es habitual que sea un proceso completamente manual (basado por lo general en técnicas estadísticas). Sin embargo, cuando la cantidad de datos de los que disponemos aumenta la resolución manual del problema se hace intratable. Aquí es donde entra en juego el conjunto de técnicas de análisis automático al que nos referimos al hablar de Minería de Datos o KDD.

Hasta ahora, los mayores éxitos en Minería de Datos se pueden atribuir directa o indirectamente a avances en bases de datos (un campo en el que los ordenadores superan a los humanos). No obstante, muchos problemas de representación del conocimiento y de reducción de la complejidad de la búsqueda necesaria (usando conocimiento a priori) están aún por resolver. Ahí reside el interés que ha despertado el tema entre investigadores de todo el mundo.

A continuación se presentan varias definiciones de Minería de Datos (MD):

- “MD es la extracción no trivial de información implícita, desconocida previamente, y potencialmente útil desde los datos” [PSF91].
- “MD es el proceso de extracción y refinamiento de conocimiento útil desde grandes bases de datos” [SLK96].
- “MD es el proceso de extracción de información previamente desconocida, válida y procesable desde grandes bases de datos para luego ser utilizada en la toma de decisiones” [CHSVZ].
- "MD es la exploración y análisis, a través de medios automáticos y semiautomáticos, de grandes cantidades de datos con el fin de descubrir patrones y reglas significativos" [BERR97].
- "MD es el proceso de planteamiento de distintas consultas y extracción de información útil, patrones y tendencias previamente desconocidas desde grandes cantidades de datos posiblemente almacenados en bases de datos” [THUR99].
- “MD es el proceso de descubrir modelos en los datos” [WF00].

1.1.4. Tecnologías de Apoyo

Para el estudio de la Minería de Datos se ha tomado la perspectiva orientada a datos, por dos razones. Primero porque la mayoría de los trabajos en Minería de Datos están enfocados hacia el *data warehouse* que proporciona el apoyo a la Minería de Datos organizando y estructurando los datos. Además, otras tecnologías de apoyo a la minería datos han sido utilizadas desde hace tiempo

y la integración de estas tecnologías con la administración de datos ha contribuido mucho a mejorar la Minería de Datos.

Las más importantes entre estas tecnologías son los métodos estadísticos [DEGR86] y el aprendizaje automático [MIT97]. Los métodos estadísticos han producido varios paquetes estadísticos [THUR99] para computar sumas, promedios, y distribuciones, que han ido integrándose con las bases de datos a explorar. El aprendizaje automático consiste en la obtención de reglas de aprendizaje y modelos de los datos, para lo cual a menudo se necesita la ayuda de la estadística. Por esta razón, los métodos estadísticos y el aprendizaje automático son los dos componentes más importantes de la Minería de Datos. Además existen otras tecnologías, entre las que se incluyen visualización, procesamiento paralelo, y apoyo a la toma de decisiones. Las técnicas de visualización ayudan a presentar los datos para facilitar la Minería de Datos. Las técnicas procesamiento paralelo ayudan a mejorar el rendimiento de la Minería de Datos. Los sistemas de apoyo a la toma de decisiones ayudan a discriminar los resultados y proporcionan los resultados esenciales para llevar a cabo las funciones de dirección.

Razonamiento estadístico

Las técnicas y métodos estadísticos del razonamiento han sido utilizados durante varias décadas, siendo los únicos medios de analizar los datos en el pasado. Numerosos paquetes [THUR99] están ahora disponibles para computar promedios, sumas, y diferentes distribuciones para diferentes aplicaciones. Por ejemplo, la oficina del censo usa análisis y métodos estadísticos para analizar la población en un país. Más recientemente, las técnicas estadísticas del razonamiento están jugando un papel importante en la Minería de Datos. Algunos paquetes estadísticos que han sido utilizados durante mucho tiempo, se han integrado con las diferentes bases de datos, y se están comercializándose en la actualidad como productos para la Minería de Datos.

La estadística juega un importante papel en el análisis de los datos, e incluso también en el aprendizaje automático. Debido a esto, no se puede estudiar la Minería de Datos sin un buen conocimiento de la estadística.

Visualización

Las tecnologías de la visualización muestran gráficamente los datos en las bases de datos. Se ha investigado mucho sobre la visualización y el campo ha adelantado un gran trecho sobre todo con la incorporación de la informática multimedia. Por ejemplo, los datos en las bases de datos serán filas y filas de valores numéricos, y las herramientas de visualización toman estos datos y trazan con ellos algún tipo de gráfico. Los modelos de visualización pueden ser bidimensionales, tridimensionales o incluso multidimensionales. Se han desarrollado varias herramientas de visualización para integrarse con las bases de datos, y algunos trabajos sobre este tema están recogidos en [VIS95].

Así, las herramientas de visualización ayudan de forma interactiva a la Minería de Datos, aunque hay pocos trabajos sobre la integración de las herramientas

de Minería de Datos y de visualización. Algunas ideas preliminares se presentaron en el *IEEE Databases and Visualization Workshop* de 1995 (véase, por ejemplo, [VIS95]). Sin embargo, se han realizado más progresos que se pueden encontrar en [VIS97], aunque queda todavía mucho trabajo por hacer en este tema.

Procesamiento paralelo

El procesamiento paralelo es una técnica que ha sido utilizado durante mucho tiempo. El área se ha desarrollado significativamente, desde sistemas con un único procesador hasta sistemas multiprocesador. Los sistemas de multiprocesamiento pueden estar formados por sistemas distribuidos o por sistemas centralizados de multiprocesadores con memoria compartida, o con multiprocesadores sin memoria compartida. Hay muchos trabajos sobre la utilización de las arquitecturas paralelas para el procesamiento de las bases de datos (véase, por ejemplo, [IEEE89]). A pesar de haberse realizado considerable trabajo sobre el tema, estos sistemas no fueron comercializados hasta el desarrollo del *data warehouse*, ya que muchos de los *data warehouses* emplean el procesamiento paralelo para acelerar el proceso de las consultas.

En un sistema de bases de datos paralelas, se ejecutan varias operaciones y funciones en paralelo. A pesar de que la investigación en sistemas de bases de datos en paralelo empezó en los años setenta, estos sistemas se han empezado a utilizar para las aplicaciones comerciales recientemente, debido en parte a la explosión del *data warehouse* y de las tecnologías de Minería de Datos dónde el rendimiento de los algoritmos de consulta es crítico. Para escalar las técnicas de Minería de Datos se necesita hardware y software apropiado, por lo que los fabricantes de bases de datos están empleando ordenadores con procesamiento paralelo para llevar a cabo la Minería de Datos.

Apoyo a la toma de decisiones

Los sistemas de apoyo a la toma de decisiones son las herramientas que usan los directivos para tomar decisiones eficaces, y se basan en la teoría de la decisión. Se puede considerar a las herramientas de Minería de Datos como tipos especiales de herramientas de apoyo a la toma de decisiones. Las herramientas de apoyo a la toma de decisiones pertenecen a una amplia categoría (véase, por ejemplo, [DECI]).

En general, las herramientas de apoyo a la toma de decisiones podrían utilizarse también como herramientas para eliminar los resultados innecesarios e irrelevantes obtenidos de la Minería de Datos. También pueden ser consideradas de este tipo, herramientas tales como las hojas de cálculo, sistemas expertos, sistemas de hipertexto, sistemas de gestión de información de web, y cualquier otro sistema que ayude a analistas y gestores a manejar eficazmente grandes cantidades de datos e información. Recientemente ha aparecido un área nueva llamada gestión del conocimiento. La gestión del conocimiento trata de manejar eficazmente los datos, la información, y el conocimiento de una organización [MORE98a].

Se puede pensar que el apoyo a la toma de decisiones es una tecnología que se solapa con la Minería de Datos, almacenamiento de datos, gestión del conocimiento, aprendizaje automático, estadística, y otras tecnologías que ayudan gestionar el conocimiento de una organización y los datos.

Aprendizaje automático

El aprendizaje automático, en muchos casos, consiste fundamentalmente en el aprendizaje de reglas a partir de los datos [MIT97], y por eso muchas de las técnicas de aprendizaje automático son utilizadas en la actualidad en la Minería de Datos.

El aprendizaje automático aparece continuamente en la realización de aprendizaje computacional desde la experiencia. Como Mitchell describe en su excelente texto sobre aprendizaje automático [MIT97], el aprendizaje automático consiste en aprender de las experiencias del pasado con respecto a alguna medida de rendimiento. Por ejemplo, en las aplicaciones de los juegos de computadora, el aprendizaje automático podría ser aprender a jugar un juego de ajedrez, desde las experiencias del pasado que podrían ser juegos que el ordenador juega contra sí mismo, con respecto a alguna medida de rendimiento, como ganar un cierto número de partidas.

Se han desarrollado distintas técnicas en el aprendizaje automático, incluyendo el aprendizaje conceptual donde se aprende los conceptos desde diferentes ejemplos de entrenamiento, las redes de neuronas, los algoritmos genéticos, los árboles de decisión, y la programación de la lógica inductiva. Se han realizado diferentes estudios teóricos sobre el aprendizaje automático, que intentan determinar la complejidad y capacidad de las diferentes técnicas de aprendizaje automático [MIT97].

Los investigadores del aprendizaje automático han agrupado las técnicas en tres categorías [THUR99]. La primera es el aprendizaje activo que se ocupa de la interacción y realización de las consultas durante el aprendizaje, la segunda es el aprendizaje desde el conocimiento anterior, y la tercera es el aprendizaje incremental. Hay alguna superposición entre los tres métodos. Durante un seminario sobre aprendizaje automático [DARP98] fueron estudiados los problemas y desafíos en aprendizaje automático y sus relaciones con la Minería de Datos. Hay todavía mucha investigación que realizar en este área, sobre todo en la integración del aprendizaje automático con las diferentes técnicas de gestión de datos. Tal investigación mejorará significativamente el área de Minería de Datos. Algunos de los algoritmos más conocidos de aprendizaje automático se encuentran en [QUIN93, MBK98].

1.1.5. Áreas de Aplicación

En este punto se presentan las principales áreas y sectores empresariales en las que se puede aplicar la minería de datos.

Marketing

Actualmente con la generación de los puntos de ventas informatizados y conectados a un ordenador central, y el constante uso de las tarjetas de créditos se genera gran cantidad de información que hay que analizar. Con ello se puede emplear la minería de datos para:

- Identificar patrones de compra de los clientes: Determinar cómo compran, a partir de sus principales características, conocer el grado de interés sobre tipos de productos, si compran determinados productos en determinados momentos,...

- Segmentación de clientes: Consiste en la agrupación de los clientes con características similares, por ejemplo demográficas. Es una importante herramienta en la estrategia de marketing que permite realizar ofertas acordes a diferentes tipos de comportamiento de los consumidores.

- Predecir respuestas a campañas de *mailing*: Estas campañas son caras y pueden llegar a ser molestas para los clientes a los que no le interesan el tipo de producto promocionado por lo que es importante limitarlas a los individuos con una alta probabilidad de interesarse por el producto. Está por ello muy relacionada con la segmentación de clientes.

- Análisis de cestas de la compra [market-basket analysis]: Consiste en descubrir relaciones entre productos, esto es, determinar qué productos suelen comprarse junto con otros, con el fin de distribuirlos adecuadamente.

Compañías de Seguros

En el sector de las compañías de seguros y la salud privada, se pueden emplear las técnicas de minería de datos, por ejemplo para:

- Análisis de procedimientos médicos solicitados conjuntamente.

- Predecir qué clientes compran nuevas pólizas.

- Identificar patrones de comportamiento para clientes con riesgo.

- Identificar comportamiento fraudulento.

Banca

En el sector bancario la información que puede almacenarse es, además de las cuentas de los clientes, la relativa a la utilización de las tarjetas de crédito, que puede permitir conocer hábitos y patrones de comportamiento de los usuarios. Esta información puede aplicarse para:

- Detectar patrones de uso fraudulento de tarjetas de crédito.

- Identificar clientes leales: Es importante para las compañías de cualquier sector mantener los clientes. Y es que hay estudios que demuestran que

es cuatro veces más caro obtener nuevos clientes que mantener los existentes.

- Predecir clientes con probabilidad de cambiar su afiliación.
- Determinar gasto en tarjeta de crédito por grupos.
- Encontrar correlaciones entre indicadores financieros.
- Identificar reglas de mercado de valores a partir de históricos.

Telecomunicaciones

En el sector de las telecomunicaciones se puede almacenar información interesante sobre las llamadas realizadas, tal como el destino, la duración, la fecha,... en que se realiza la llamada, por ejemplo para:

- Detección de fraude telefónico: Mediante por ejemplo el agrupamiento o *clustering* se pueden detectar patrones en los datos que permitan detectar fraudes.

Medicina

También en el campo médico se almacena gran cantidad de información, sobre los pacientes, tal como enfermedades pasadas, tratamientos impuestos, pruebas realizadas, evolución,...

Se pueden emplear técnicas de minería de datos con esta información, por ejemplo, para:

- Identificación de terapias médicas satisfactorias para diferentes enfermedades.
- Asociación de síntomas y clasificación diferencial de patologías.
- Estudio de factores (genéticos, precedentes, hábitos, alimenticios,...) de riesgo para la salud en distintas patologías.
- Segmentación de pacientes para una atención más inteligente según su grupo.
- Estudios epidemiológicos, análisis de rendimientos de campañas de información, prevención, sustitución de fármacos,...
- Identificación de terapias médicas y tratamientos erróneos para determinadas enfermedades.

Industria farmacéutica

En el sector químico y farmacéutico se almacenan gran cantidad de información:

- Bases de datos de dominio público conteniendo información sobre estructuras y propiedades de componentes químicos.

- Resultados de universidades y laboratorios publicadas en revistas técnicas.

- Datos generados en la realización de los experimentos.

- Datos propios de la empresa.

Los datos son almacenados en diferentes categorías y a cada categoría se le aplica un diferente trato. Se podrían realizar, entre otras, las siguientes operaciones con la información obtenida:

- *Clustering* de moléculas: Consiste en el agrupamiento de moléculas que presentan un cierto nivel de similitud, con lo que se pueden descubrir importantes propiedades químicas.

- Búsqueda de todas las moléculas que contienen un patrón específico: Se podría introducir una subestructura (un patrón), devolviendo el sistema todas las moléculas que son similares a dicha estructura.

- Búsqueda de todas las moléculas que vincula un camino específico hacia una molécula objetivo: Realizar una búsqueda exhaustiva puede ser impracticable, por lo que se pueden usar restricciones en el espacio de búsqueda.

- Predicción de resultado de experimentos de una nueva molécula a partir de los datos almacenados: A través de determinadas técnicas de inteligencia artificial es posible predecir los resultados a nuevos experimentos a partir de los datos, con el consiguiente ahorro de tiempo y dinero.

Biología

Con la finalización en los próximos años del Proyecto Genoma Humano y el almacenamiento de toda la información que está generando en bases de datos accesibles por Internet, el siguiente reto consiste en descubrir cómo funcionan nuestros genes y su influencia en la salud. Existen nuevas tecnologías (chips de ADN, proteómica, genómica funcional, variablidad genética individual) que están posibilitando el desarrollo de una “nueva biología” que permite extraer conocimiento biomédicos a partir de bases de datos experimentales en el entorno de un ordenador básicamente mediante técnicas de minería de datos y visualización. Estos trabajos forman parte de los desarrollos de la Bioinformática.

1.1.6. Tendencias de la Minería de Datos

El interés que despierta la Minería de Datos para el análisis de la información especialmente en el área comercial hace que se busquen nuevas aplicaciones basadas en esta tecnología. Algunas de las principales nuevas aplicaciones basadas en la Minería de Datos se presentan a continuación.

Minería de Textos

La Minería de Textos [Text Mining] surge ante el problema cada vez más apremiante de extraer información automáticamente a partir de masas de textos. Se trata así de extraer información de datos no estructurados: texto plano.

Existen varias aproximaciones a la representación de la información no estructurada [HH96]:

- "Bag of Words": Cada palabra constituye una posición de un vector y el valor corresponde con el número de veces que ha aparecido.
- *N*-gramas o frases: Permite tener en cuenta el orden de las palabras. Trata mejor frases negativas "*... excepto ...*", "*... pero no ....*", que tomarían en otro caso las palabras que le siguen como relevantes.
- Representación relacional (primer orden): Permite detectar patrones más complejos (si la palabra *X* está a la izquierda de la palabra *Y* en la misma frase...).
- Categorías de conceptos.

Casi todos se enfrentan con el "vocabulary problem" [FUR87]: Tienen problemas con la sinonimia, la polisemia, los lemas, etc.

Un ejemplo de aplicación basada en Minería de Textos es la generación automática de índices en documentos. Otras más complicadas consistirían en escanear completamente un texto y mostrar un mapa en el que las partes más relacionadas, o los documentos más relacionados se coloquen cerca unos de otros. En este caso se trataría de analizar las palabras en el contexto en que se encuentren.

En cualquier caso, aunque aún no se ha avanzado mucho en el área de Minería de Textos, ya hay productos comerciales que emplean esta tecnología con diferentes propósitos.

Minería de datos Web

La Minería de datos Web [Web Mining] es una tecnología usada para descubrir conocimiento interesante en todos los aspectos relacionados a la Web. Es uno de los mayores retos. El enorme volumen de datos en la Web generado por la explosión de usuarios y el desarrollo de librerías digitales hace que la extracción de la información útil sea un gran problema. Cuando el usuario

navega por la web se encuentra frecuentemente saturado por los datos. La integración de herramientas de minería de datos puede ayudar a la extracción de la información útil.

La Minería de datos Web se puede clasificar en tres grupos distintos no disjuntos, dependiendo del tipo de información que se quiera extraer, o de los objetivos [KB00]:

- Minería del *Contenido* de la Web [Web Content Mining]: Extraer información del contenido de los documentos en la web. Se puede clasificar a su vez en:
    - Text Mining: Si los documentos son textuales (planos).
    - Hypertext Mining: Si los documentos contienen enlaces a sí mismos o a otros documentos
    - Markup Mining: Si los documentos son semiestructurados (con marcas).
    - Multimedia Mining: Para imágenes, audio, vídeo,...
- Minería de la Estructura de la Web [Web Structure Mining]: Se intenta descubrir un modelo a partir de la tipología de enlaces de la red. Este modelo puede ser útil para clasificar o agrupar documentos.
- Minería del Uso de la Web [Web Usage Mining]: Se intenta extraer información (hábitos, preferencias, etc. de los usuarios o contenidos y relevancia de documentos) a partir de las sesiones y comportamiento de los usuarios navegantes

1.2. Minería de Datos y Almacenamiento de Datos

Como se ha enfatizado repetidamente, los datos son críticos para hacer data mining. Por consiguiente, se necesitan sistemas de bases de datos para manejar los datos a los que aplicar data mining eficazmente. Estos sistemas podrían ser sistemas de data warehouse o sistemas de bases de datos.

1.2.1. Arquitectura, Modelado, Diseño, y Aspectos de la Administración

Las técnicas de data mining existen desde hace algún tiempo. ¿Por qué entonces data mining se ha hecho tan popular ahora? La principal razón es que ahora con los sistemas de bases de datos se pueden representar, almacenar y recuperar los datos, y reforzar características como la integridad y seguridad.

Ahora que se tienen los datos guardados en las bases de datos y quizás normalizados y estructurados, ¿Cómo se puede hacer data mining? Un enfoque es reforzar un SGBD con una herramienta de data mining. Se puede comprar un SGBD comercial y una herramienta de data mining comercial que tenga construidas las interfaces para el SGBD y se puede aplicar la herramienta a los datos administrados por el SGBD. A pesar de que este enfoque tiene ventajas y promueve las arquitecturas abiertas, hay algunos inconvenientes. Podría haber algunos problemas de rendimiento cuando se usa un SGBD de propósito general para data mining.

El otro enfoque es una integración fuerte del SGBD con las herramientas de data mining. El núcleo de la base de datos tiene las herramientas de data mining incorporadas dentro de él. Se puede decir que este tipo de SGBD es un Mining SGBD (SGBD de data mining). Según esto las diferentes funciones del SGBD como el procesamiento de consultas y la gestión del almacenamiento son influenciadas por las técnicas de data mining. Por ejemplo, los algoritmos de optimización pueden ser modificados por las técnicas de data mining. Se ha investigado mucho sobre la integración de data mining y el núcleo del SGBD (véase [TSUR98]).

Mining SGBD también significaría la eliminación de funciones innecesarias de un SGBD y el protagonismo de las características clave. Por ejemplo, el procesamiento de transacciones es una función soportada por la mayoría de los SGBD comerciales. Sin embargo, data mining normalmente no se dirige a los datos transaccionales sino a los datos de apoyo a la toma de decisiones. Estos datos no pueden ser datos que se actualicen a menudo por transacciones. Así que, podrían eliminarse funciones como la gestión de transacciones en un Mining SGBD, y se podría dar más importancia a las características adicionales que proporcionen integridad y calidad a los datos.

En el general, en el caso de un Mining SGBD, la agregación de una herramienta de data mining influirá sobre las diferentes funciones del SGBD como: el procesamiento de consultas, la gestión del almacenamiento, la gestión de transacciones, la gestión de metadata (diccionario de datos), la gestión de la seguridad y de la integridad.

El tipo de modelado de los datos usado puede tener algún impacto en data mining. Muchos de los datos que serán utilizados se guardan en bases de datos relacionales. Sin embargo, actualmente cada vez más se guardan los datos en bases de datos no relacionales tales como bases de datos orientadas a objetos, bases de datos objeto-relacionales y bases de datos multimedia. Hay poca información sobre data minig en bases de datos orientadas a objetos, aunque si hay algunos trabajos sobre data mining en las bases de datos multimedia. En las bases de datos orientadas a objetos primero se extraen las relaciones entre los objetos y se guardan en una base de datos relacional, y después las herramientas de data mining se aplican a la base de datos relacional.

El diseño de la base de datos juega un papel fundamental en la aplicación de data mining. Por ejemplo, en el caso de data warehousing, se han propuesto diferentes enfoques en el modelo y subsiguiente diseño del almacén. Éstos

incluyen modelos multidimensionales de datos y modelos del procesamiento analítico en línea. Se han propuesto varios esquemas como el esquema en estrella para el almacenamiento de los datos. Como se ha mencionado, la organización eficaz de los datos es crítica para data mining. Por consiguiente también, tales modelos y esquemas son importantes para data mining

La administración de las bases de datos también resulta influida por la realización de data mining. Si se integra data mining un SGBD, aparecen las siguientes cuestiones ¿Con qué frecuencia será aplicado data mining a la base de datos? ¿Puede ser usado data mining para analizar la auditoria de datos? ¿Como influirá en data mining la actualización frecuente de los datos? Éstas interesantes preguntas tendrán respuestas cuando se obtenga más información sobre la integración de data mining con las funciones del SGBD.

1.2.2. Data mining y Funciones de Bases de datos

En el caso de integración fuerte entre el SGBD y data mining hay un fuerte impacto sobre las diferentes funciones del sistema de bases de datos. Por ejemplo, en el procesamiento de consultas. Se han realizado trabajos para examinar lenguajes de consultas como SQL y determinar si se necesitan extensiones para soportar data mining (véase por ejemplo [ACM96a]). Si hay estructuras adicionales y consultas que son complejas, entonces el optimizador de consultas tiene que ser adaptado para manejar esos casos. Estrechamente relacionado con la optimización de consultas esta la eficiencia de las estructuras de almacenamiento, índices, y métodos de acceso. Pueden ser necesarios mecanismos especiales para apoyar data mining en el procesamiento de consultas.

En el caso de gestión de transacciones, la realización de data mining puede tener poco impacto, puesto que data mining se hace normalmente en los datos de apoyo a la toma de decisiones y no en los datos transaccionales. Sin embargo hay casos dónde se analizan los datos transaccionales para anomalías como en los casos de tarjetas de crédito y de tarjetas de teléfono. A veces las compañías de tarjetas de crédito o de teléfono han notificado sobre usos anómalos de tarjetas de crédito o de teléfono. Esto normalmente se hace analizando los datos transaccionales. También se podría aplicar data mining a estos datos.

En el caso de metadata, se podría aplicar data mining a metadata para extraer la información útil en casos dónde los datos no sean analizables. Ésta puede ser la situación para datos no estructurados cuyo metadata deba ser estructurado. Por otro lado, los metadata podrían ser un recurso muy útil para una herramienta de data mining. Metadata podría dar información adicional para ayudar con el proceso de data mining.

La seguridad, integridad, calidad del datos, y tolerancia a fallos son influidas por data mining. En el caso de seguridad, data mining podría suponer una amenaza importante para la seguridad y privacidad.

Por otro lado data mining pueden usarse para descubrir las intrusiones así como para analizar la auditoria de datos. En el caso de auditoria, la cantidad de datos sobre los que se aplica data mining es grande. Se pueden aplicar las herramientas de data mining a los datos para descubrir los modelos anormales. Por ejemplo, si un empleado hace un excesivo número de viajes a un país determinado y este hecho es conocido, proponiendo algunas preguntas. La siguiente pregunta a realizar es si el empleado tiene asociaciones con ciertas personas de ese país. Si la respuesta es positiva, entonces la conducta del empleado se marca.

Como ya se ha mencionado data mining tiene muchas aplicaciones en el descubrimiento de la intrusión y analizando amenazas a las bases de datos. Se puede usar data mining para descubrir modelos de intrusiones y amenazas. Ésta es un área emergente y se llama Información de Confianza. No sólo es importante tener datos de calidad, también es importante recuperarse de fallos maliciosos o de otro tipo, y proteger los datos de amenazas o intrusiones. Aunque la investigación en esta área simplemente está empezando, se esperan grandes progresos.

En el caso de calidad e integridad de los datos, se podrían aplicar las técnicas de data mining para descubrir datos malos y mejorar la calidad de los datos. Data mining también pueden usarse para analizar la seguridad de los datos para varios sistemas como sistemas de control de circulación aérea, sistemas nuclear, y sistemas de armamento.

1.2.3. DATA WAREHOUSE

Un **data warehouse** es un tipo especial de base de datos. Al parecer, el término se originó a finales de los ochenta [DEVL88], [INMO88], aunque el concepto es más antiguo. La referencia [INMO93] define un data warehouse como "un almacén de datos orientado a un tema, integrado, no volátil y variante en el tiempo, que soporta decisiones de administración" (donde el término no *volátil* significa que una vez que los datos han sido insertados, no pueden ser cambiados, aunque sí pueden ser borrados). Los data warehouses surgieron por dos razones: primero, la necesidad de proporcionar una fuente única de datos limpia y consistente para propósitos de apoyo para la toma de decisiones; segundo, la necesidad de hacerlo sin afectar a los sistemas operacionales.

Por definición, las cargas de trabajo del data warehouse están destinadas para el apoyo a la toma de decisiones y por lo tanto, tienen consultas intensivas (con actividades ocasionales de inserción por lotes); asimismo, los propios data warehouses tienden a ser bastante grandes (a menudo mayores que 500GB y con una tasa de crecimiento de hasta el 50 por ciento anual). Por consecuencia, es difícil -aunque no imposible- perfeccionar el rendimiento. También puede ser un problema la escalabilidad. Contribuyen a ese problema (a) los errores de diseño de la base de datos, (b) el uso ineficiente de los operadores relacionales, (e) la debilidad en la implementación del modelo relacional del DBMS, (d) la falta de escalabilidad del propio **DBMS** y (e) los errores de diseño arquitectónico que limitan la capacidad e imposibilitan la escalabilidad de la plataforma.

- **DATA MARTS**

Los usuarios a menudo realizaban amplias operaciones de informes y análisis de datos sobre un subconjunto relativamente pequeño de todo el data warehouse. Asimismo, era muy probable que los usuarios repitieran las mismas operaciones sobre el mismo subconjunto de datos cada vez que era actualizado. Además, algunas de esas actividades -por ejemplo, análisis de pronósticos, simulación, modelado de datos de negocios del tipo "qué pasaría si..."- involucraban la creación de nuevos esquemas y datos con actualizaciones posteriores a esos nuevos datos.

La ejecución repetida de tales operaciones sobre el mismo subconjunto de todo el almacén no era muy eficiente; por lo tanto, pareció buena idea construir algún tipo de "almacén" limitado de propósito general que estuviera hecho a la medida de ese propósito. Además, en algunos casos sería posible extraer y preparar los datos requeridos directamente a partir de las fuentes locales, lo que proporcionaba un acceso más rápido a los datos que si tuvieran que ser sincronizados con los demás datos cargados en todo el data warehouse. Dichas consideraciones condujeron al concepto de **data marts.**

De hecho, hay alguna controversia sobre la definición precisa del término *data mart.* Se puede definir como "un almacén de datos especializado, orientado a un tema, integrado, volátil y variante en el tiempo para apoyar un subconjunto específico de decisiones de administración". La principal diferencia entre un data mart y un data warehouse es que el data mart es *especializado y volátil. Especializado* quiere decir que contiene datos para dar apoyo (solamente) a un área específica de análisis de negocios; por *volátil* se entiende que los usuarios pueden actualizar los datos e incluso, posiblemente, crear nuevos datos (es decir, nuevas tablas) para algún propósito.

Hay tres enfoques principales para la creación de un data mart:

- Los datos pueden ser simplemente extraídos del data warehouse; se sigue un enfoque de "divide y vencerás" sobre la carga de trabajo general de apoyo para la toma de decisiones, a fin de lograr un mejor rendimiento y escalabilidad. Por lo general, los datos extraídos son cargados en una base de datos que tiene un esquema físico que se parece mucho al subconjunto aplicable del data warehouse; sin embargo, puede ser simplificado de alguna manera gracias a la naturaleza especializada del data mart.

- A pesar del hecho de que el data warehouse pretende proporcionar un "punto de control único", un data mart puede ser creado en forma independiente (es decir, no por medio de la extracción a partir del data warehouse). Dicho enfoque puede ser adecuado si el data warehouse es inaccesible por alguna causa: razones financieras, operacionales o incluso políticas (o puede ser que ni siquiera exista todavía el data warehouse).

- Algunas instalaciones han seguido un enfoque de "primero el data mart", donde los data marts son creados conforme van siendo necesarios y el data warehouse general es creado, finalmente, como una consolidación de los diversos data marts.

Los últimos dos enfoques sufren posibles problemas de desacople semántico. Los data marts independientes son particularmente susceptibles a tales problemas, debido a que no hay forma obvia de verificar los desacoples semánticos cuando las bases de datos son diseñadas en forma independiente. Por lo general, la consolidación de data marts en data warehouses falla, a menos que (a) se construya primero un esquema lógico único para el data warehouse y (b) los esquemas para los data marts individuales se deriven después a partir del esquema del data warehouse.

Un aspecto importante en el diseño de data marts: es la **granularidad** de la base de datos. Donde *granularidad* se refiere al nivel más bajo de agregación de datos que se mantendrá en la base de datos. Ahora bien, la mayoría de las aplicaciones de apoyo para la toma de decisiones requerirán tarde o temprano acceso a datos detallados y por lo tanto, la decisión será fácil para el data warehouse. Para un data mart puede ser más difícil. La extracción de grandes cantidades de datos detallados del data warehouse, y su almacenamiento en el data mart, puede ser muy ineficiente si ese nivel de detalle no se necesita con mucha frecuencia. Por otro lado, en algunas ocasiones es difícil establecer definitivamente cuál es el nivel más bajo de agregación que en realidad se necesita. En dichos casos, los datos detallados pueden ser accedidos directamente desde el data warehouse cuando se necesiten, manteniendo en el data mart los datos que de alguna manera ya fueron agregados.

- **APLICACIONES DE LOS DATA WAREHOUSE**

La explotación del Data Warehouse puede realizarse mediante diversas técnicas:

- Query & Reporting
- On-line analytical processing (OLAP)
- Executive Information System (EIS)
- Decision Support Systems (DSS)
- Visualización de la información
- Data Mining ó Minería de Datos, etc.

Se llaman sistemas OLAP (On Line Analytical Processing) a aquellos sistemas que deben:

- Soportar requerimientos complejos de análisis
- Analizar datos desde diferentes perspectivas

- Soportar análisis complejos contra un volumen ingente de datos

La funcionalidad de los sistemas OLAP se caracteriza por ser un análisis multidimensional de datos mediante navegación del usuario por los mismos de modo asistido.

Existen dos arquitecturas diferentes para los sistemas OLAP: OLAP multidimensional (MD-OLAP) y OLAP relacionales (ROLAP).

La arquitectura MD-OLAP usa bases de datos multidimensionales, la arquitectura ROLAP implanta OLAP sobre bases de datos relacionales

La arquitectura MD-OLAP requiere unos cálculos intensivos de compilación. Lee de datos precompilados, y tiene capacidades limitadas de crear agregaciones dinámicamente o de hallar ratios que no se hayan precalculado y almacenado previamente.

La arquitectura ROLAP, accede a los datos almacenados en un Data Warehouse para proporcionar los análisis OLAP. La premisa de los sistemas ROLAP es que las capacidades OLAP se soportan mejor contra las bases de datos relacionales.

Los usuarios finales ejecutan sus análisis multidimensionales a través del motor ROLAP, que transforma dinámicamente sus consultas a consultas SQL. Se ejecutan estas consultas SQL en las bases de datos relacionales, y sus resultados se relacionan mediante tablas cruzadas y conjuntos multidimensionales para devolver los resultados a los usuarios. ROLAP es una arquitectura flexible y general, que crece para dar soporte a amplios requerimientos OLAP. El MOLAP es una solución particular, adecuada para soluciones departamentales con unos volúmenes de información y número de dimensiones más modestos.

Una cuestión típica de un sistema OLAP o DSS podría ser: “¿Compraron más monovolúmenes en 1998 los habitantes del norte de España, o los del sur?” Sin embargo, un sistema data mining en este escenario podría ser interrogado así:

“Quiero un modelo que identifique las características predictivas más importantes de las personas que compran monovolumenes...”

- **QUERY & REPORTING**

Las consultas o informes libres trabajan tanto sobre el detalle como sobre las agregaciones de la información.

Realizar este tipo de explotación en un almacén de datos supone una optimización del tradicional entorno de informes (reporting), dado que el Data Warehouse mantiene una estructura y una tecnología mucho más apropiada para este tipo de solicitudes.

Los sistemas de "Query & Reporting", no basados en almacenes de datos se caracterizan por la complejidad de las consultas, los altísimos tiempos de

respuesta y la interferencia con otros procesos informáticos que compartan su entorno.

1.2.4. DATA WAREHOUSE Y DATA MINING

Data warehouse almacena los datos de las bases de datos heterogéneas para que los usuarios consulten sólo un único aspecto. Las respuestas que un usuario consigue a una consulta dependen de los volúmenes del data warehouse. El data warehouse en general no intenta extraer la información de los datos almacenados. Data warehouse estructura y organiza los datos para suportar funciones de administración, data mining intenta extraer la información útil, así como predecir las tendencias de los datos. La Figura 3 10 ilustra la relación entre el data warehouse y data mining. Observe que no es necesario construir un data warehouse para hacer data mining, ya que también puede aplicarse data mining a las bases de datos. Sin embargo, un data warehouse estructura los datos de tal manera que facilita data mining, por lo que en muchos casos es muy deseable tener un almacén del datos para llevar a cabo data mining..

¿Dónde acaba data warehouse y donde empieza data mining? ¿Hay una diferencia clara entre data warehouse y data mining? La respuesta es subjetiva. Hay ciertas preguntas que los data warehouse pueden contestar. Además, los data warehouse disponen de capacidades para el apoyo a la toma de decisiones. Algunos data warehouse llevan a cabo predicciones y tendencias. En este caso los data warehouse llevan a cabo algunas de las funciones de data mining. En el general, en el caso de un data warehouse la respuesta está en la base de datos. El data warehouse tiene que disponer de optimización de consultas y técnicas de acceso para obtener respuestas. Por ejemplo, considere preguntas como ¿"Cuántos automóviles rojos compraron los médicos en 1990 en Nueva York "? La respuesta está en la base de datos. Sin embargo, para una pregunta como " ¿Cuántos automóviles rojos comprarán los médicos en 2005 en Nueva York "? la respuesta no puede estar en la base de datos. Basándose en los patrones de compra de los médicos en Nueva York y sus proyecciones del sueldo, se podría predecir la respuesta a esta pregunta.

Esencialmente, un warehouse organiza los datos eficazmente para realizar data mining sobre ellos. La pregunta es entonces ¿Es imprescindible tener un warehouse para hacer data mining? La respuesta es que es muy interesante tener un warehouse, pero esto no significa que sea imprescindible. Podría usarse un buen SGBD para gestionar una base de datos eficazmente. También, a menudo con un warehouse no se tienen datos transaccionales. Por lo tanto, los datos no pueden ser actuales, y los resultados obtenidos desde data mining tampoco lo serán. Si se necesita la información actualizada, entonces se podría hacer data mining sobre una base de datos administrada por un SGBD que también tenga características de procesamiento de transacciones. Hacer data mining sobre datos que se actualizan a menudo es un desafío. Típicamente data mining se ha usado sobre los datos de apoyo a la toma de decisiones. Por consiguiente hay varios problemas que necesitan ser investigados extensamente, antes de que se pueda llevar a cabo lo que se conoce como data mining en tiempo real. De momento al menos, es crítico

disponer de un buen data warehouse para llevar a cabo un buen data mining para funciones de apoyo a la toma de decisiones. Observe que también se podría tener una herramienta integrada para llevar a cabo las funciones de data warehouse y data mining. Una herramienta de este tipo será conocida como data warehouse miner.

1.3. Herramientas Comerciales de Análisis de Datos

KnowledgeSeeker de Angoss Software International, Toronto, Canada

Puntos Clave:

- Herramienta interactiva de clasificación.
- Basada en los algoritmos de árboles de decisión CHAID y XAID.
- Se ejecuta sobre plataformas Windows y UNIX

Ventajas:

- Representación flexible de árboles de decisión.
- Provee características para permitir la identificación de la relevancia de los resultados en los negocios.
- El API permite usar los resultados del análisis en aplicaciones personalizadas.

Aspectos a tener en cuenta:

- Solo soporta árboles de decisión
- Poco soporte para la transformación de datos.
- El soporte para predicción se limita a la exportación de las reglas generadas.

Cuando usarla:

- Si se necesita una herramienta que permita adelantar una visión instantánea general de sus datos.
- Si necesita una herramienta interactiva para explorar sus datos.
- No está indicada si se necesita una herramienta que soporte predicción desde dentro de sus datos.

DataCruncher de DataMind, San Mateo, CA, USA

Puntos Clave:

- Herramienta de Data Mining para clasificación y clustering
- Basada en Tecnología de agentes de redes (ANT Agent Network Technology)
- La aplicación servidor se ejecuta sobre UNIX y Windows NT; la aplicación cliente en todas las plataformas Windows.

Ventajas:

- Fácil de usar, ya que los modelos necesitan pocas adaptaciones.
- Agent Network Technology puede ser utilizada para clasificación, predicción y clustering no supervisado.
- Resultados versátiles, que permiten una minuciosa valoración de los modelos y de sus resultados

Aspectos a tener en cuenta:

- Se necesita familiarizarse con la tecnología para comprender los resultados.
- Está basada en una técnica propietaria
- Tiene soporte limitado para la transformación de datos.

Cuando usarla:

- Si se necesita una herramienta cliente-servidor con una interface fácil de usar.
- Si se necesita valorar para cada caso la bondad de la predicción de los modelos.
- Si quiere invertir algún esfuerzo en hacer un completo uso del análisis de resultados.

Intelligent Miner de IBM, Armonk, NY, USA

Puntos Clave:

- Soporta múltiples operaciones de data mining en un entrono cliente-servidor

- Utiliza redes de neuronas, árboles de inducción y varias técnicas estadísticas.
- Trabaja sobre clientes Windows, OS/2 y X-Windows, y servidores AIX (incluyendoSP2), OS/400 y OS/390.

Ventajas:

- Buen soporte para análisis de asociaciones y clustering (incluyendo visualización de clustering), además de clasificación y predicción.
- Optimizada para data mining en grandes bases de datos(del orden de gigabytes) ya que se aprovecha de la plataforma de procesamiento paralelo PS2 de IBM.
- Tiene un entorno de trabajo integrado con características muy interesantes tanto para usuarios expertos como no especialistas.

Aspectos a tener en cuenta:

- Algunos problemas que tenía han sido resueltos con la nueva interface que ha sido desarrollada completamente en Java.
- Solo trabaja sobre plataformas IBM, y el acceso a los datos se limita a las bases de datos DB2 y a ficheros planos.
- Inicialmente la mayoría de los proyectos requerirán entradas importantes desde los servicios de soporte y consultoría de IBM

Cuando usarla:

- Debería ir a una tienda de IBM para observar la funcionalidad del data mining integrado en su entorno de soporte a las decisiones
- Para grandes proyectos de data mining, en particular cuando los datos están contenidos en DB2.
- Si se desan utilizar varias operaciones de data mining, tales como clasificación, clustering y análisis de asociaciones.
- Para realizar análisis de cesta de la compra con varios gigabytes de datos.
- Si interesa utilizar los servicios de consultoría de IBM.

Clamentine de Integral Solutions, Basingstoks, UK

Puntos Clave:

- Herramienta con un entrono de trabajo que soporta todo el proceso de data mining
- Ofrece árboles de decisión, redes de neuronas, generación de reglas de asociación y características de visualización.
- Se ejecuta sobre VMS, UNIX o Windows NT.

Ventajas:

- Interface gráfica intuitiva para programación visual.
- Las técnicas de data mining pueden complementarse combinándose entre si.
- Visión interactiva de las relaciones entre las variables a través de grafos de red.

Aspectos a tener en cuenta:

- No soporta Windows nativo.
- Es necesario familiarizarse con la herramienta para conseguir una óptima utilización de sus funcionalidades.
- No está optimizada para arquitecturas en paralelo.

Cuando usarla:

- Si se necesita una herramienta que cubra por completo el rango de los procesos de data mining.
- Si se desean combinar herramientas y modelos para construir los procesos de data mining que exijan tales requisitos.
- Si se desea desarrollar el modelo en C.
- Si se necesitan grandes capacidades analíticas y de gestión de datos sin requerir un extenso análisis de datos ni experiencia en tecnologías informáticas.

Alice de Isoft SA, Gif sur Yvette, Francia.

Puntos Clave:

- Herramienta de escritorio para data mining interactivo.
- Se basa en tecnología de árboles de decisión.
- Se ejecuta sobre plataformas Windows.

Ventajas:

- La representación altamente interactiva permite guiar el análisis.
- La opción de generar gráficos provee una visión general de los datos en todas las etapas del proceso de Data Mining.
- Se trata de una herramienta económica valida para usuarios que comienzan a realizar data mining.

Aspectos a tener en cuenta:

- No tiene opciones para desarrollar modelos.
- Pequeño soporte para transformación de datos.
- No genera conjuntos de reglas optimizadas desde los árboles de decisión.

Cuando usarla:

- Si se desea usar data mining para buscar patrones y relaciones en los datos.
- Si se quiere tener la posibilidad de dirigir el análisis interactivamente.
- Si no se es un experto en data mining y se desea realizar el análisis.
- Si se quiere entender los patrones que se encuentran en la base de datos y no se desea construir modelos predictivos.

Decisión Series, de NeoVista Software Cupertino CA, USA.

Puntos Clave:

- Herramientas para múltiples operaciones de data mining para el desarrollo de modelos basados en servidores.
- Proporciones algoritmos de redes de neuronas, árboles y reglas de inducción, clustering y análisis de asociaciones.
- Trabaja sobre sistemas UNIX mono o multi-procesadores de HP y Sun. Accede sólo a ficheros planos, aunque posiblemente las últimas versiones ya trabajaran contra bases de datos relacionales.

Ventajas:

- Soporta un gran rango de operaciones y algoritmos de data mining, la mayoría de los cuales han sido altamente optimizados para obtener altos rendimientos.

- Está optimizado para plataformas que trabajan en paralelo con grandes conjuntos de datos.
- Ofrece una considerable flexibilidad para construir modelos de alto rendimiento para aplicaciones de usuario final embebidas.

Aspectos a tener en cuenta:

- Las herramientas de desarrollo gráfico son bastante básicas.
- Poco soporte para la exploración de datos.
- La mayoría de los clientes necesitaran un considerable soporte de consultas para generar aplicaciones y ejecutarlas. Es necesario tener conocimientos de análisis de datos y de utilización de UNIX para desarrollar las aplicaciones.

Cuando usarla:

- Si se desean construir aplicaciones con alto rendimiento de modelos de data mining embebidos que utilizan entornos con multiprocesadores.
- Si se quiere tener un absoluto control sobre todos los elementos de los procesos de construcción de modelos.
- Si se necesitan combinar operaciones y tecnicas de data mining alternativas en aplicaciones complejas.
- Si se quiere trabajar con una solución que puede comunicar una aplicación data mining para enlazar con sus necesidades.

Pilot Discovery Server de Pilot Software, Cambridge MA, USA.

Puntos Clave:

- Herramienta para clasificación y predicción.
- Basada en la tecnología de árboles de decisión CART.
- Trabaja sobre UNIX y Windows NT

Ventajas:

- Buena representación del análisis de resultados
- Es fácil de usar y de entender.
- Muy integrada con sistemas gestores de bases de datos relacionales.

Aspectos a tener en cuenta:

- Solamente indicada para clientes de los programas para soporte a la toma de decisiones de Pilot.
- Solamente cubre un especifico sector del espectro del data mining.
- Sólo trabaja con datos almacenados en bases de datos relacionales.

Cuando usarla:

- Si se desea optimizar las campañas de marketing.
- Si se necesita interpretar fácilmente los resultados sin realizar un gran refinamiento de los modelos.
- Solo si se están utilizando los programas para soporte a la toma de decisiones de Pilot.
- No está indicada si se quieren resolver los problemas utilizando diferentes técnicas.

SAS Solution for Data Mining de SAS Institute, Cary, NC, USA

Puntos Clave:

- Un gran número de herramientas de selección, exploración y análisis de datos para entornos cliente-servidor.
- Las opciones de data mining incluyen: aplicaciones de redes de neuronas, de árboles de decisión y herramientas de estadística.
- Aplicaciones portables para un gran número de entornos PC, UNIX y mainframes.

Ventajas:

- SAS ofrece data warehouse y análisis de datos.
- Conjuntos extensibles de herramientas de manipulación y visualización de datos.
- SAS tiene una gran experiencia en herramientas estadísticas y de análisis de datos.

Aspectos a tener en cuenta:

- La oferta para hacer data mining es una mezcolanza de todas las técnicas SAS existentes.
- Integración con la programación en 4GL.

- No soporta el análisis de asociaciones.

Cuando usarla:

- Si ya se utiliza SAS para almacenar, administrar y analizar los datos.
- Si se va a utilizar SAS para la construcción del data warehouse.
- Si es necesaria una alta funcionalidad en la manipulación de datos.
- Si se es experto en estadística y se quieren utilizar las funciones estadísticas de SAS.

MineSet, de Silicon Graphics, Mountain View, CA, USA

Puntos Clave:

- Paquete de herramientas para Data mining y visualización.
- Proporciona algoritmos para la generación de reglas para clasificación y asociaciones.
- Trabaja sobre plataformas SGI bajo IRIS.

Ventajas:

- Ofrece herramientas de visualización para los datos y los modelos generados.
- Suporta muchas operaciones de data mining.
- El gestor de herramientas actúa como un punto central de control y permite el acceso y transformación de los datos.

Aspectos a considerar:

- Requiere un servidor SGI.
- La gran cantidad de opciones y parámetros puede provocar confusión en usuarios noveles.
- Las herramientas de visualización necesitan mucha preparación y personalización de los datos para producir buenos resultados.

Cuando usarla:

- Si se quieren detectar patrones por visualización.
- Si se quieren construir aplicaciones que representen los resultados de data mining a través de visualización.

- Si se dispone de equipos de Silicon Graphics
- Esta indicada para VARs que quieran desarrollar soluciones personalizadas de data mining usando MineSet.

SPSS, de SPSS, Chicago IL, USA

Puntos Clave:

- Herramientas de escritorio para clasificación y predicción, clustering, y un gran rango de operaciones estadísticas.
- Proporciona una herramienta de redes de neuronas además de productos de análisis estadístico.
- SPSS para Windows y Neural Connection son productos que trabajan en modo monopuesto en plataformas Windows.

Ventajas:

- Las funciones de análisis estadístico complejo son accesibles a través de una interface de usuario muy bien diseñada.
- Neural Connection ofrece un amplio rango de opciones y funciones a través un entorno de desarrollo muy fácil de usar.
- El lenguaje de scripts permite una gran personalización del entorno y el desarrollo de aplicaciones estadísticas aisladas.

Aspectos a considerar:

- Para analistas de datos y estadísticos, más que para usuarios finales.
- SPSS CHAID carece de la funcionalidad de otros productos de escritorio de árboles de decisión.
- Neural Connection es un producto aislado: la base de la integración con SPSS es a través de transferencia de datos, que se limita a la importación de 32.000 registros.

Cuando usarla:

- Si se necesita un análisis complejo combinando estadística con árboles de decisión y redes de neuronas.
- Si se disponen de grandes conocimientos estadísticos y se quiere utilizar data mining basado en IA.
- Si se necesita verificación estadística de los resultados encontrados.

- Si es preciso construir aplicaciones de análisis departamental para escritorio.
- Si tiene un presupuesto ajustado.

Syllogic Data Mining Tool, de Syllogic, Houten, The Netherlands

Puntos Clave:

- Herramienta con entorno de trabajo multi-estratégico con interface visual.
- Soporta análisis de árboles de decisión, clasificación k-vecino más próximo, y análisis de clustering y asociaciones por k-means.
- Trabaja sobre Windows NT y en estaciones UNIX con uno o varios procesadores

Ventajas:

- La interface visual permite a los usuarios construir proyectos de data mining enlazando objetos.
- La versión está optimizada para entornos masivamente paralelos y validos para grandes bases de datos.
- La empresa también ofrece un gran número de servicios de consultaría en las áreas de datawarehousing y data mining.

Aspectos a considerar:

- La interface y la presentación de resultados necesita algunos refinamientos para ser utilizada por usuarios finales.
- DMT/MP no soportan el mismo rango de operaciones que DMT

Cuando usarla:

- Si se necesita servicio de consultoría a la vez que se desarrolla el proyecto de data mining con un entorno de datawarehousing.
- Si se necesita utilizar gran número de operaciones de data mining.
- Si se quiere utilizar una herramienta similar en el escritorio y en el entorno MP.

Darwin de Thinking Machines, Bedford MA, USA

Puntos Clave:

- Herramientas de desarrollo de data mining de tipo cliente-servidor para la construcción de modelos de clasificación y predicción.
- La construcción de modelos utiliza algoritmos de redes de neuronas, árboles de inducción y k-vecino más próximo.
- Trabaja sobre plataformas Sun de Solaris, AIX de IBM y SP2, con clientes Motif. También existen versiones cliente que trabajan sobre Windows.

Ventajas:

- Ofrecen buena cobertura al proceso completo de descubrimiento del conocimiento.
- Pone el énfasis en el desarrollo de modelos predictivos de alto rendimiento.
- Proporciona escalabilidad para soportar paralelización.

Aspectos a considerar:

- Mejor para analistas de datos y desarrolladores de aplicaciones que para los usuarios de negocio.
- Es preciso familiarizarse con las diferentes opciones de Darwin para cada tipo de modelo si se quiere obtener el mejor resultado de la herramienta.
- No soporta análisis no supervisado de clustering o de asociaciones.

Cuando usarla:

- En la construcción de aplicaciones de data mining para gestión de relaciones entre clientes.
- Si se necesita una herramienta que ponga mucho énfasis en modelado por clasificación y predictivos.
- Si se dispone de una gran compleja base de datos que precise la potencia de una plataforma con multiprocesadores.
- Si se necesita observar la creación de los modelos de data mining, Darwin proporciona múltiples algoritmos y varias opciones de refinamiento.
- Si se quiere usar las herramientas de data mining para auxiliar la gestión de redes Thinking Machina tiene objetivos muy explícitos en este sector y ya colabora con Cabletron.

1.4. Arquitectura Software para Data Mining

Anteriormente se han discutido diferentes tecnologías para data mining. Se necesita el apoyo arquitectónico para integrar estas tecnologías. La Figura 1.4 muestra una pirámide que presenta la estructura de cómo las diferentes tecnologías encajan entre si. Como se muestra en esta figura, en el nivel más bajo se encuentra las comunicaciones y sistemas. A continuación aparece el soporte del middleware. Esto va seguido por la gestión de la bases de datos y el data warehouse. Después aparecen las diferentes tecnologías de data mining. Finalmente, se tienen los sistemas de apoyo a la toma de decisiones que usan los resultados de data mining y ayudan a que los usuarios tomen las decisiones eficazmente. Estos usuarios pueden ser administradores, analistas, programadores, y cualquier otro usuario del sistema de información.

Cuando se construyen sistemas, las diferentes tecnologías involucradas pueden no encajar exactamente en la pirámide tal como se ha mostrado. Por ejemplo, se podría saltar la fase de data warehouse y se podría ir directamente a la herramienta de data mining. Uno de los problemas importantes, en este punto, son las interfaces entre los diferentes sistemas. En la actualidad no se tiene bien definida cualquiera de las interfaces normales excepto en el caso de algunos de los lenguajes estándar de definición de interfaz que surgen de los diferentes grupos como el Object Management Group. Sin embargo, cuando estas tecnologías vayan madurando, se irán desarrollando los estándares para las interfaces.



Figura 1.4: *Pirámide para Data mining*

Ya se ha estudiado cómo las diferentes tecnologías trabajan juntas. Por ejemplo, una posibilidad es la mostrada en la Figura 1.5 donde se integran múltiples bases de datos a través de algún middleware y como consecuencia forman un data warehouse que se explora a continuación. Los componentes de data mining también se integran en este escenario para aplicar data mining a

las bases de datos directamente. Algunos de estos problemas se discutirán en la sección de la arquitectura del sistema.

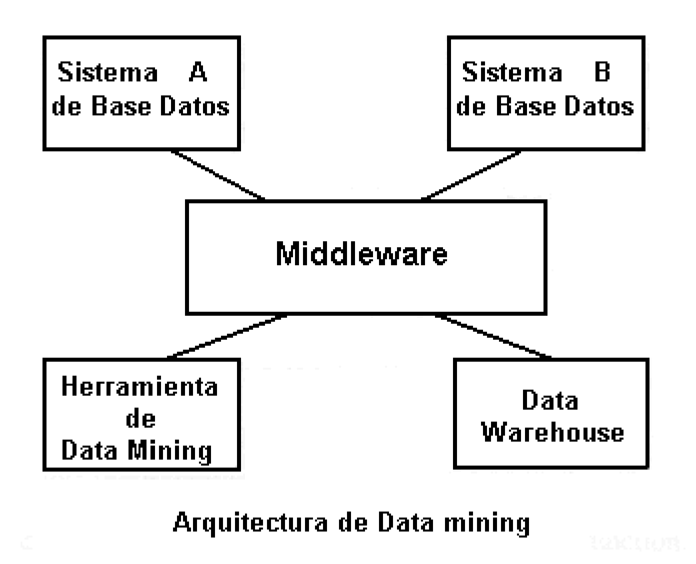


Figura 1.5: *Arquitectura de data mining*

La figura 1.6 ilustra una vista tridimensional de las tecnologías de data mining. En el centro se encuentra la tecnología para la integración. Ésta es la tecnología del middleware tal como la gestión distribuida orientada al objeto y también la tecnología web para la integración y acceso a través de web.

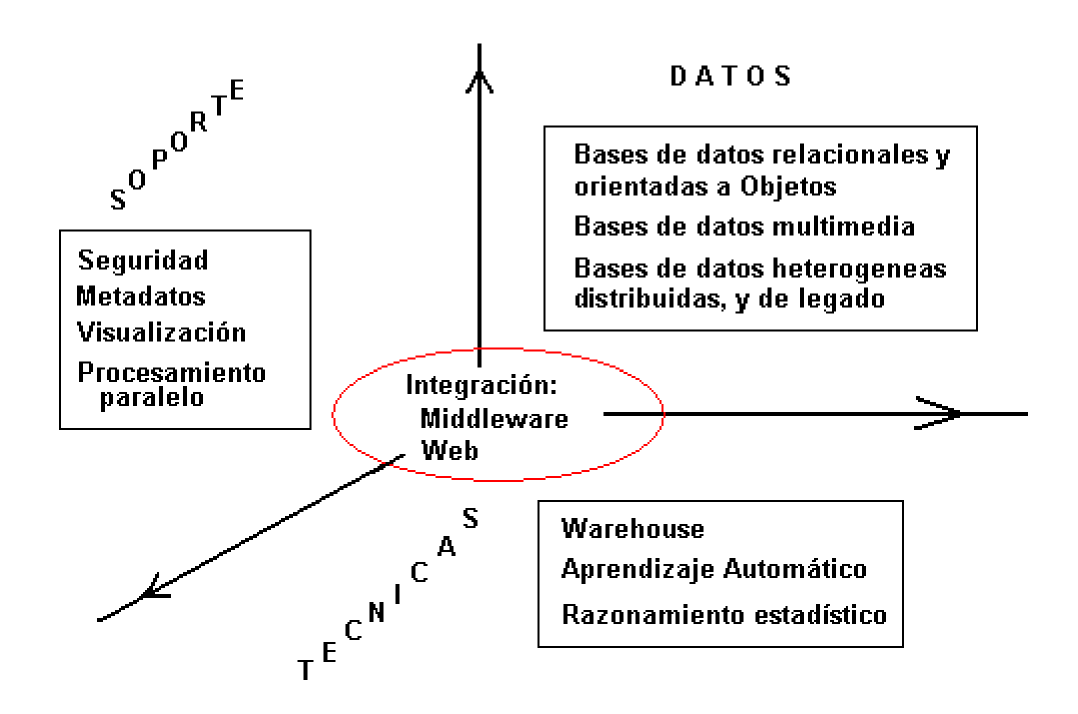


Figura 1.6: *Visión en tres dimensiones*

En una primera dimensión tenemos todas las tecnologías básicas de datos como multimedia, bases de datos relacionales y orientadas a objetos, y bases de datos distribuidas, heterogéneas y de herencia. En la segunda dimensión tenemos las tecnologías para realizar data mining. Aquí se ha incluido el warehousing así como el aprendizaje automático, tal como la programación de la lógica inductiva, y el razonamiento estadístico. La tercera dimensión comprende tecnologías como el procesamiento paralelo, la visualización, gestión de metadatos (diccionario de datos), y el acceso seguro que son importantes para llevar a cabo data mining.

1.4.2. Arquitectura Funcional

A continuación se describen los componentes funcionales de data mining. Anteriormente se discutieron los componentes funcionales de un sistema de gestión de bases de datos. En adición, se mostro una arquitectura en la que la herramienta de data mining era uno de los módulos del SGBD. Un SGBD con estas características será un SGBD Mining. Un SGBD Mining se puede organizar de varias maneras. Un enfoque alternativo se ilustra en Figura 4. En este enfoque se considera data mining como una extensión del procesador de consultas. Es decir, podrían extenderse los módulos del procesador de consultas como el optimizador de consultas para ocuparse de data mining. Esto es una vista de alto nivel como se ilustra en la Figura 1.7. Observe que en este diagrama se ha omitido al gestor de las transacciones, ya que data mining se usa principalmente en el procesamiento analítico en línea (OLTP).

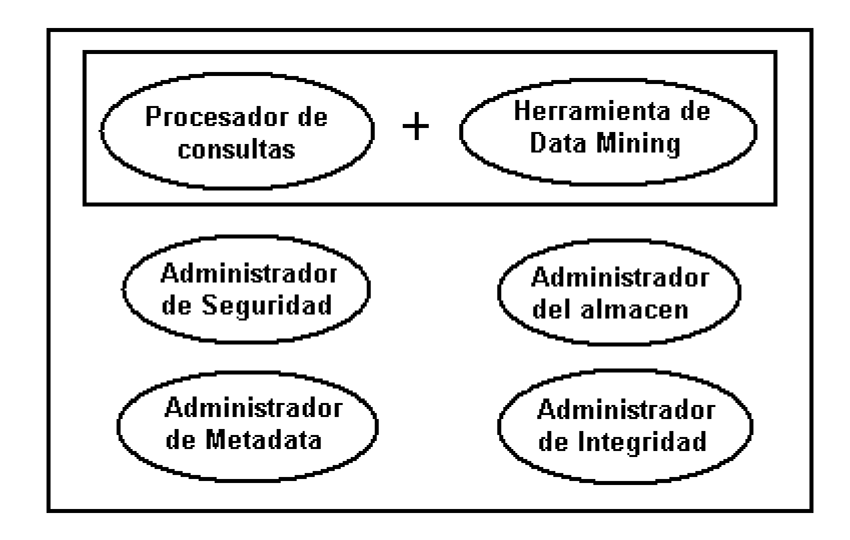


Figura 1.7: *Data mining como parte del procesador de consultas*

La pregunta es: ¿Cuáles son los componentes de la herramienta de data mining? Como se ilustra en la Figura 1.8, una herramienta de data mining podría tener los siguientes componentes: un componente de aprendizaje de experiencia que usa varios conjuntos de entrenamiento y aprende varias estrategias, un componente analizador de datos que analiza los datos en base a lo que tiene que aprender, y un componente productor de resultados que realiza la clasificación, el clustering, y otras tareas como las asociaciones. Hay

interacción entre los tres componentes. Por ejemplo, el componente que produce los resultados entrega los resultados obtenidos al componente de entrenamiento para ver si este componente tiene que ser adaptado. El componente de entrenamiento da la información al componente analizador de datos. El componente de analizador de datos da la información al componente productor de los resultados.

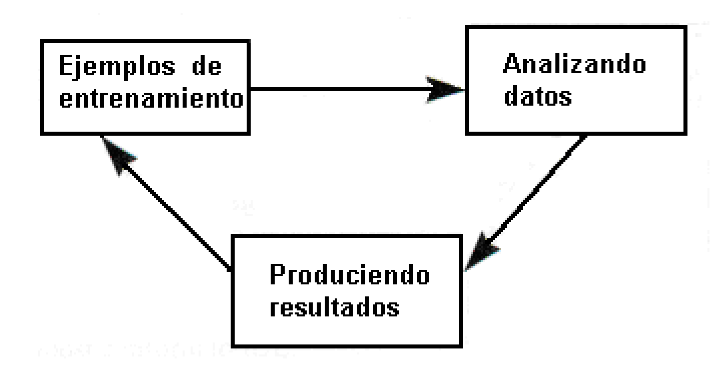


Figura 1.8: *Las Funciones de data mining*

Observe que no se han incluido componentes tales como el preprocesador de datos y el podador (refinador) de los resultados en los módulos de data mining. Estos componentes también son necesarios para completar el proceso entero. El preprocesador de datos formatea los datos. De alguna forma el data warehouse puede hacer esta función. El componente de poda o recorte de resultados puede extraer sólo la información útil. Esto podría llevarse a cabo por un sistema de apoyo a la toma de decisiones. Todos estos pasos se integrarán en el proceso de data mining.

1.4.3. Arquitectura del Sistema

Algunas de las arquitecturas que se han discutido anteriormente así como la observada en la Figura 1.5 pueden considerarse como una arquitectura del sistema para data mining. Una arquitectura del sistema consiste en componentes como los middleware y otros componentes del sistema como el sistema de bases de datos y el sistema de data warehouse para data mining.

Los middleware que se ilustran en Figura 1.5 podrían basarse en diferentes tecnologías. Un sistema middleware muy popular es el que se basa en una arquitectura cliente-servidor.

En efecto, muchos de los sistemas de bases de datos se basan en la arquitectura cliente-servidor. Middleware también incluye de facto estándares como el Open DataBase Connectivity Connectivity (ODBC) de Microsoft o sistemas distribuidos basados en objetos.

En [THUR97] se proporciona una discusión detallada de tecnologías cliente-servidor. En particular se discute el paradigma de cliente-servidor así como una apreciación global de ODBC y los sistemas de gestión distribuida de objetos como el Object Manegement Group’s (OMG) Common Object Request Broquer

Architecture (CORBA). Aquí se discute data mining con respecto al paradigma del cliente-servidor.

La mayoría de los vendedores de sistemas de bases de datos han migrado a una arquitectura llamada arquitectura de cliente-servidor. Con este enfoque, múltiples clientes acceden a los diferentes servidores de las bases de datos a través de alguna red. Una visión de alto nivel de la comunicación cliente-servidor de se ilustra en la Figura 1.9. El objetivo último es comunicar múltiples clientes vendedores con múltiples servidores vendedores de una manera transparente.

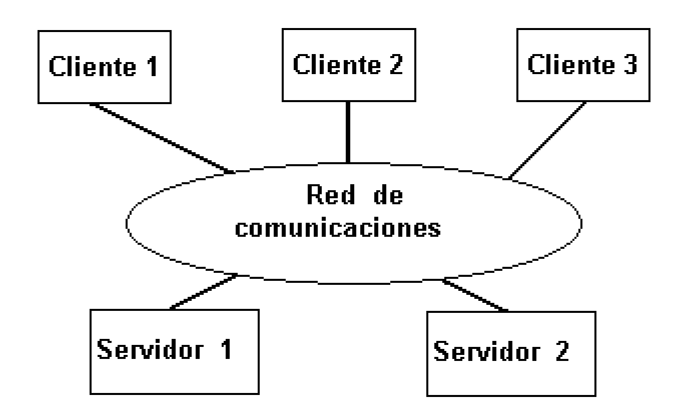


Figura 1.9: *La Arquitectura cliente-servidor de Basada en la Interoperabilidad*

En orden a facilitar la comunicación entre múltiples clientes y servidores, se han propuesto varios estándares. Un ejemplo es la Organización Internacional de Estándares (ISO), el estándar Remote Database Access (RDA). Esta norma provee una interfaz genérica para la comunicación entre un cliente y un servidor. Microsoft ODBC también ha aumentado su popularidad para la comunicación de los clientes con los servidores. El CORBA de OMG mantiene las especificaciones para las comunicaciones cliente-servidor basadas en la tecnología orientada a objetos. Aquí, una posibilidad es encapsular las bases de datos servidoras como objetos y distribuir las peticiones apropiadas de los clientes y acceder los servidores a través de un Object Request Broker (ORB). Otros estándares incluyen el DRDA de IBM (Distribuited Relational Database Access - *el Acceso de la base de datos relacional Distribuida*) y el SQL Access Group (ahora parte del Open Group); Call Level Interface *la Interfaz de Nivel de Llamada* (CLI). Se han publicado varios libros sobre computación cliente-servidor y administración de datos. Dos buenas referencias son [ORFA94] y [ORFA96]. También se estudian en detalle algunos de estos problemas en [THUR97].

Un sistema de middleware que está aumentando su popularidad para conectar sistemas heterogéneos es el CORBA de OMG. Como se declara en [OMG95], hay tres componentes principales en CORBA. Uno es el modelo orientado a objetos, el segundo es Object Request Broker *el Corredor de Demanda de Objeto* (ORB) a través del cual los clientes y servidores se comunican entre sí, y el tercero es Interface Definition Language *el Lenguaje de Definición de Interfaces* (IDL) qué específica las interfaces para la comunicación cliente-servidor. La Figura 1.10 ilustra la comunicación cliente-servidor a través de

ORB. Aquí, los clientes y servidores están encapsulados como objetos. Los dos objetos comunican entonces entre sí. La comunicación se hace mediante ORB. Además, las interfaces deben ajustarse a IDL.

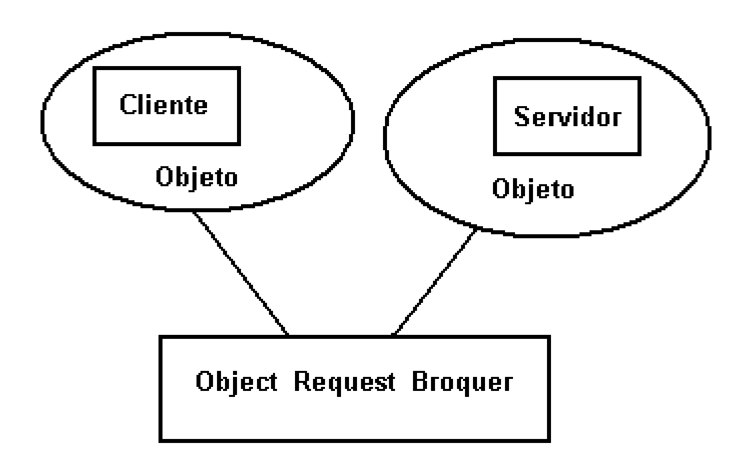


Figura 1.10: *La interoperabilidad a través del ORB*

1.4.4. El Data Mining en la Arquitectura del Sistema

Considere la arquitectura de la Figura 8. En este ejemplo, la herramienta de data mining podría usarse como un servidor, los sistemas de administración de bases de datos podrían ser otro servidor, mientras el data warehouse sería un tercer servidor. El cliente emite las peticiones al sistema de base de datos, al warehouse, y al componente de data mining como se ilustra en la figura 1.11.

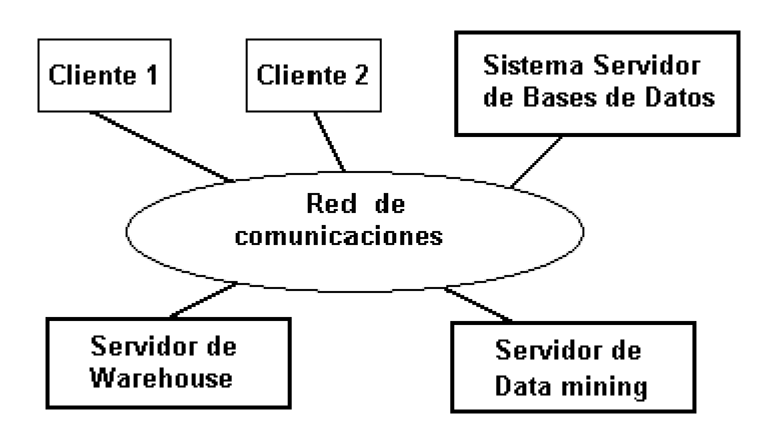


Figura 1.11: *Data mining basado en Cliente-Servidor*

También se podría usar un ORB para data mining. En este caso la herramienta de data mining se encapsula como un objeto. El sistema de bases de datos y warehouse también son objetos. Esto se ilustra en la Figura 1.12. El desafío aquí es definir IDLs para varios objetos.

Obsérvese que la tecnología cliente-servidor no desarrolla algoritmos para la administración de datos, para warehousing, o para la realización de data

mining. Esto significa que todavía se necesitan los algoritmos para realizar data mining, warehousing, y administración de la base de datos. La tecnología cliente-servidor y, en particular, la tecnología de administración distribuida de objetos como CORBA, es la que facilita la ínteroperación entre los diferentes componentes. Por ejemplo, el sistema data mining, el sistema de base de datos, y warehose comunican entre sí y con los clientes a través del ORB.

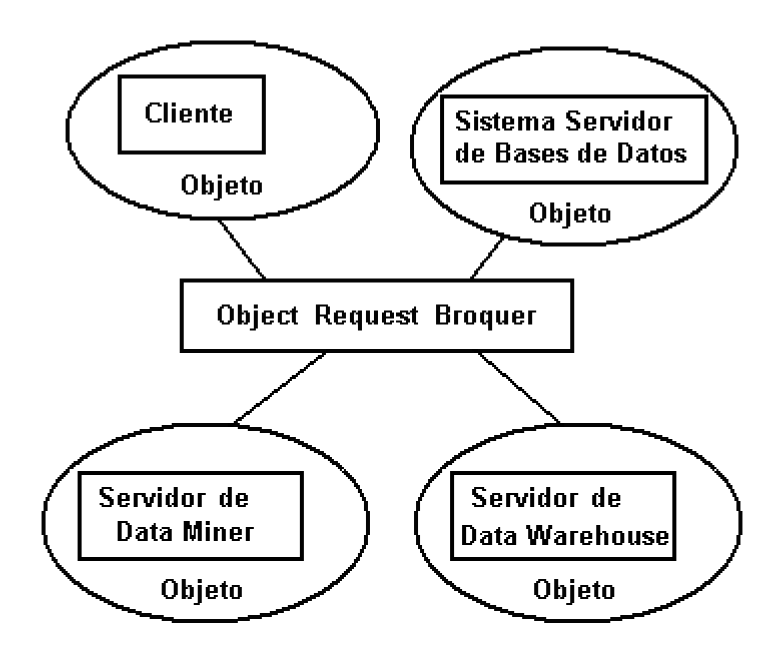


Figura 1.12: *Data mining mediante ORB*

La arquitectura a tres niveles se ha hecho muy popular (vea la discusión en [THUR971). En esta arquitectura, el cliente es un cliente ligero y realiza un procesamiento mínimo, el servidor hace las funciones de administración de la base de datos, y el nivel intermedio lleva a cabo varias funciones de proceso de negocio. En el caso de data mining, se podría utilizar también una arquitectura de tres niveles donde la herramienta de data mining se pone en el nivel intermedio. La herramienta de data mining podría desarrollarse como una colección de componentes. Estos componentes podrían estar basados en la tecnología orientada al objeto. Desarrollando los módulos de data mining como una colección de componentes, se podrían desarrollar herramientas genéricas y entonces se podría personalizarlas para las aplicaciones especializadas.

Otra ventaja de desarrollar un sistema de data mining como una colección de componentes es que se podrían comprar los componentes a vendedores diferentes y después ensamblarlos para formar un sistema. Además, los componentes podrían ser reutilizados. Por ahora asumiremos que los módulos son el integrador de los datos fuente, la herramienta de data mining, el podador (discriminador) de los resultados, y el generador de informes. Entonces cada uno de estos módulos puede encapsularse como un objeto y se podría usar ORB's para integrar estos objetos diferentes. Como resultado, se puede usar un enfoque plug-and-play en el desarrollo de herramientas de data mining. También se podría descomponer la herramienta de data mining en múltiples módulos y encapsular estos módulos como objetos. Por ejemplo, considere los módulos de la herramienta de data mining ilustrados en la Figura 5. Estos

módulos son parte del módulo de la herramienta de data mining y pueden ser encapsulados como objetos e integrados a través de un ORB.

Capítulo 2. Análisis Estadístico mediante Excel

Introducción. Métodos clásicos de análisis de datos

Descripción de datos. Estadísticos de una variable

Generalización. Distribuciones de probabilidad e intervalos de confianza

Contrastes de hipótesis. Tipos

Relaciones entre atributos

Nominales- Numéricos: Tests de comparación de medias (muestras dependientes e independientes) y análisis de varianza.

Nominales-Nominales: Tablas de Contingencia. Tests de independencia y comparación de proporciones.

Numéricos - Numéricos: Análisis de Regresión

Aplicación de técnicas estadísticas a la clasificación. Técnicas clásicas de clasificación y predicción

Clasificación mediante regresión numérica

Clasificador bayesiano

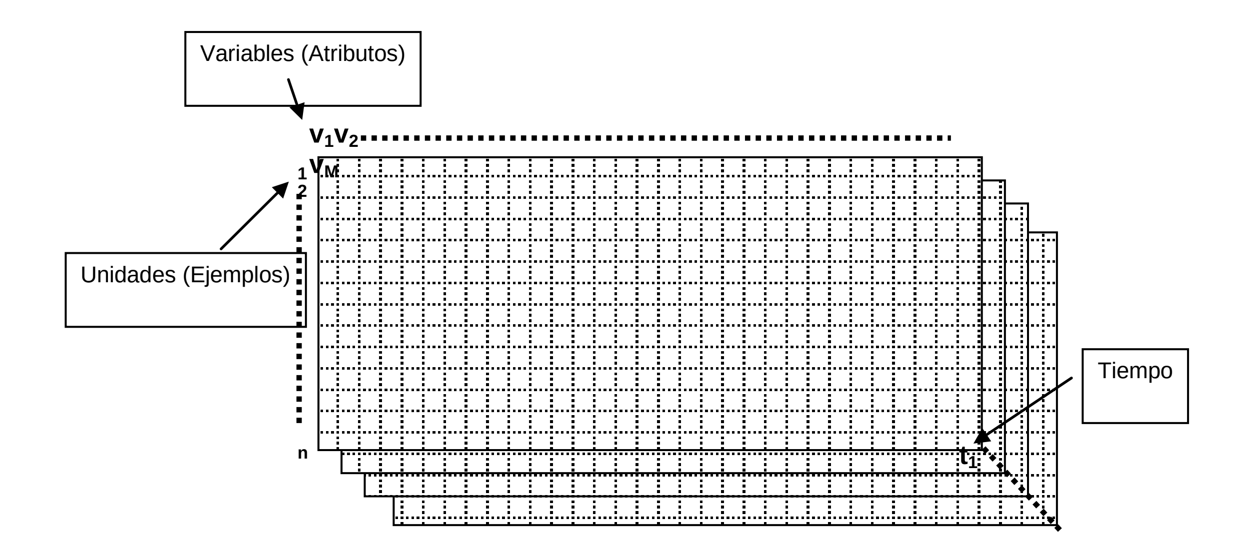
Evaluación de Hipótesis

Objetivo: se pretende validar o rechazar ideas preconcebidas a partir del análisis de los datos disponibles, generalizando las conclusiones

Pasos:

1. Generación de hipótesis
2. Determinar qué datos son necesarios. Recolectar y preparar
3. Evaluación de hipótesis para aceptar o rechazar

Matriz de datos



Tipos de variables

- nominales o categóricas (incluyendo ordinales)
- numéricas

2.1. Análisis de una variable. Estadística Descriptiva e Inferencia

- **Estadísticos:** resumen (describen) toda la información contenida en una muestra de datos :
  - Variables continuas
    - medidas centrales (media, moda, mediana)
    - medidas de dispersión (rango, varianza, desviación estándar, percentiles)
    - medidas de forma (histograma)
  - Variables nominales
    - frecuencias relativas (probabilidades), moda
    - media y varianza de probabilidad estimada
- **Muestra**: y_i; i =1…n; toma valores en un rango continuo/discreto

Estadísticos de variable continua

- **Media** (esperanza) muestral: promedio de todos los valores

$$media(y) = \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

- **Moda**: valor que aparece más veces
- **Mediana**: valor que deja el mismo número de casos a ambos lados

$$mediana(y) = y_i \mid N^o \,\text{casos}(\text{y}_\text{j} \le y_i) = N^o \,\text{casos}(\text{y}_\text{k} \ge y_i)$$

  - equivale a ordenar el vector de datos y tomar el valor central
  - menos sensible frente a valores extremos poco probables

- **Recorrido (rango):**

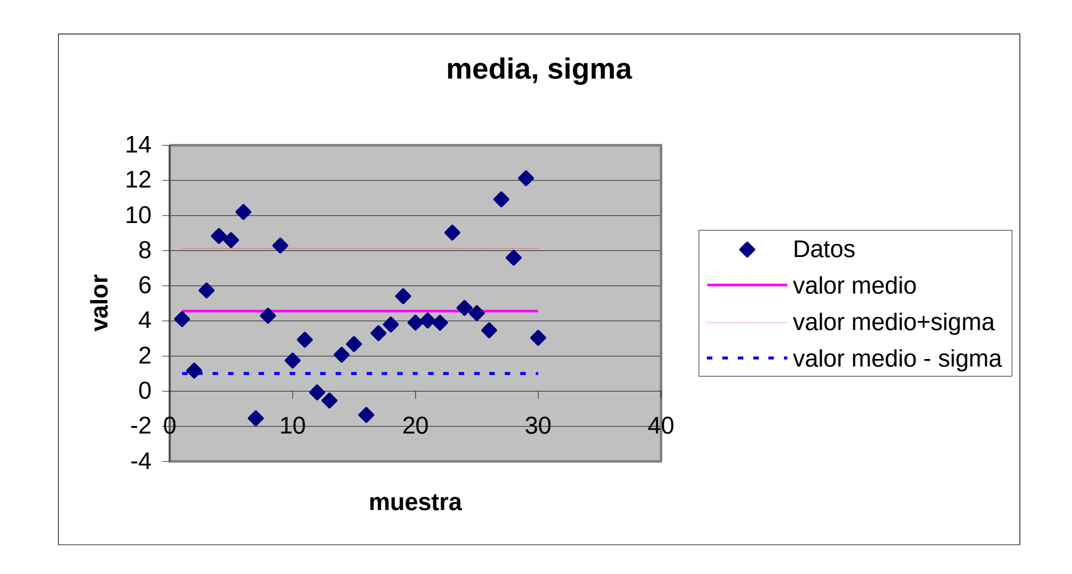
  $\max(y_i)-\min(y_i)$

- **Varianza:** promedio de desviaciones con respecto a valor medio

$$Var(y) = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2 = \frac{1}{n-1}\left[\sum_{i=1}^{n} y_i^{\,2} - n\bar{y}^2\right]$$

- **Desviación estándar** (típica): raíz cuadrada de la varianza

$$desv(y) = \sigma_y = \sqrt{Var(y)}$$

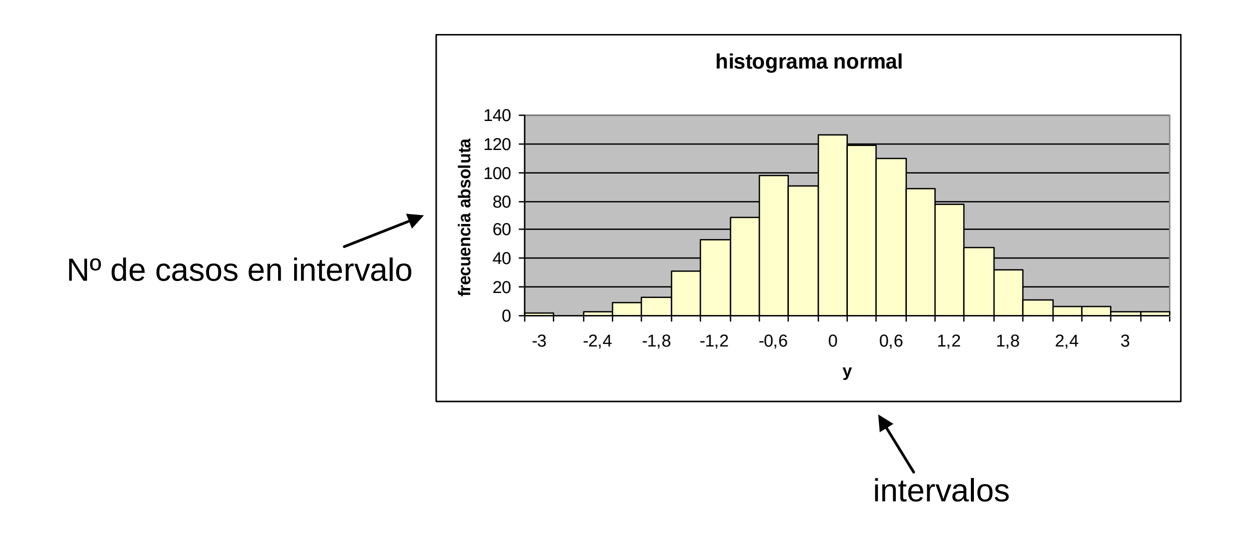


Histograma

Estimación de la distribución de densidad de probabilidad: frecuencia relativa de valores de y_i por unidad de intervalo

la suma total de frecuencias absolutas es el número de datos

la suma de frecuencias relativas es 1

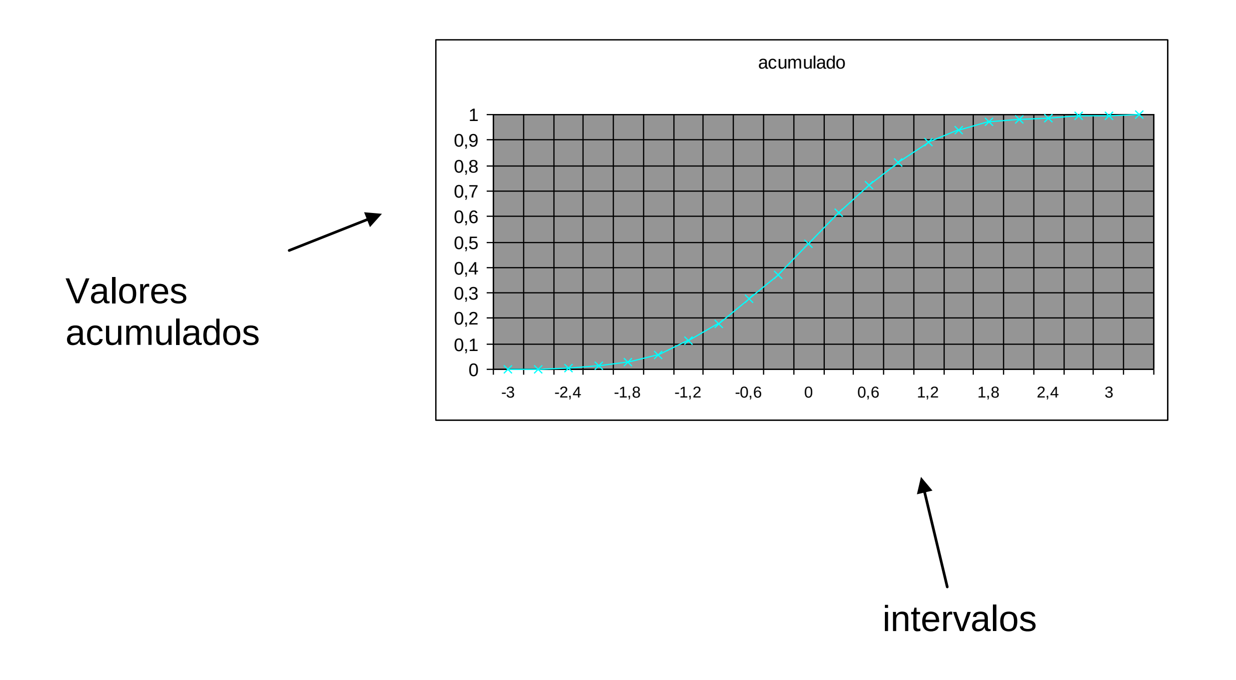


Histograma acumulado

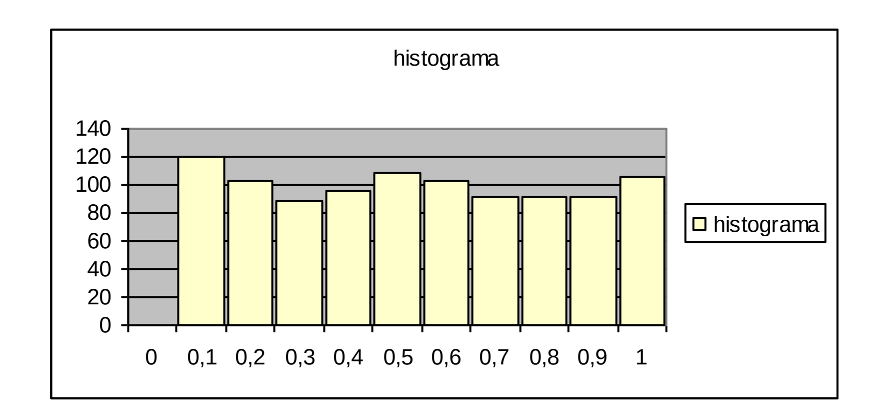
Suma de frecuencias relativas de casos inferiores al valor en abscisas (acumulación de histograma normalizado):

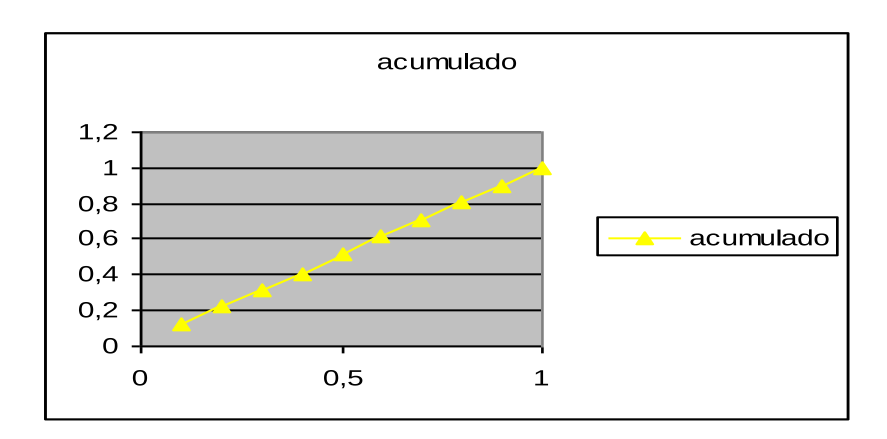
Estimación de Prob(Y<=y_i)

en el extremo superior debe ser 1



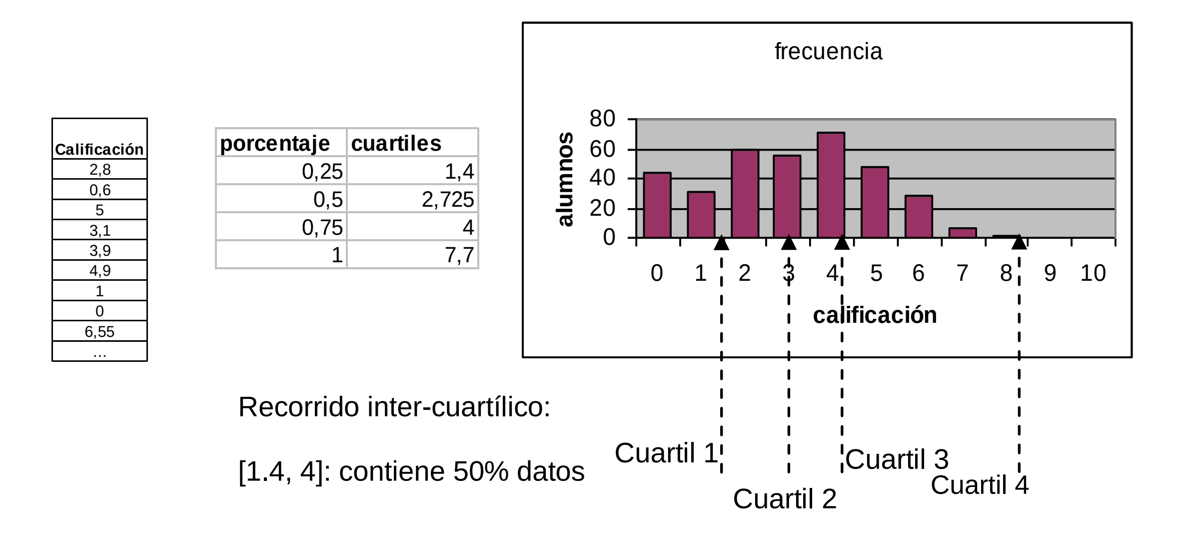
Ejemplo: histograma de variable uniforme





Cuantiles del histograma

- **Cuantil:** valores que dividen el recorrido de datos en k partes de la misma frecuencia (percentiles: 100 partes, cuartiles: 4 partes, etc.)
- Ejemplo: cuartiles



Percentiles e histograma acumulado

- Percentil p: valor que deja debajo al p% de los individuos, y al (100-p)% por encima: se entra en eje vertical del histograma acumulado

- percentil 50: mediana (por definición)

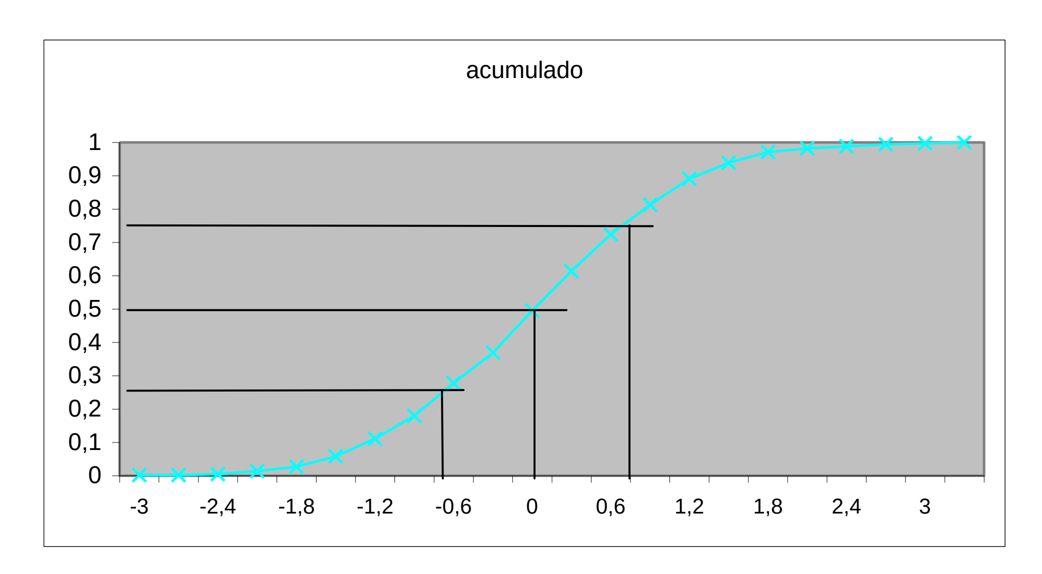
- percentiles 25, 75: *cuartiles.* Abarcan al 50% de los individuos

(recorrido inter-cuartílico)

- con distribución normal tipificada

  - percentiles 25, 75: [-0.674, 0.674]

  - percentiles 2.5, 97.5: [-1.96, 1.96]



Estadísticos de variable nominal

- y_i nominal: toma valores de un conjunto discreto (categorías): $\{v_{i1}, \ldots, v_{iki}\}$
- **Distribución de frecuencias** de cada valor

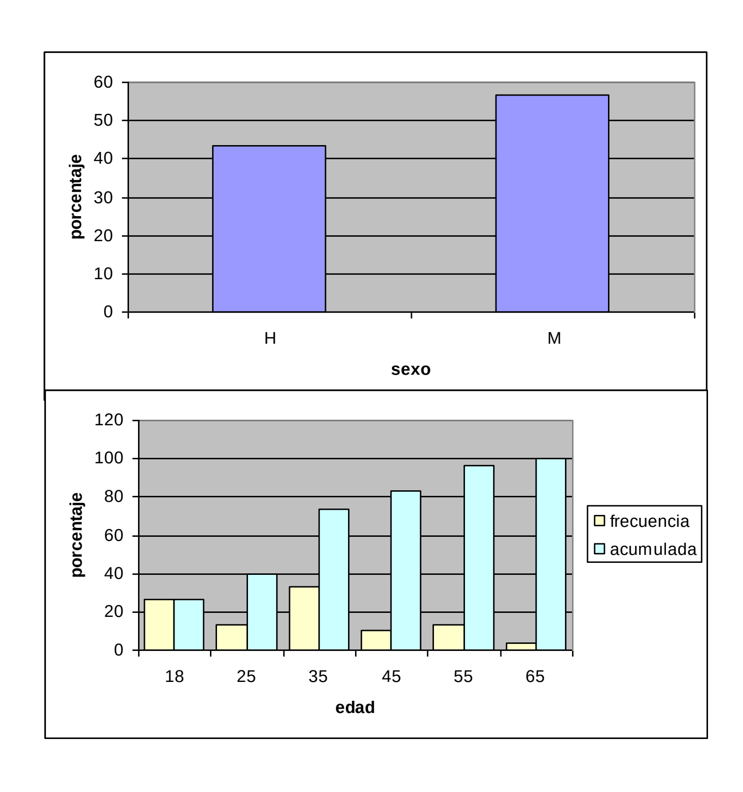
$$p_1 = 100(n_1 / n)\%$$
$$p_2 = 100(n_1 / n)\%$$
$$\vdots$$
$$p_{ki} = 100(n_{ki} / n)\%$$
$$n = \sum_{j=1}^{k_i} n_j$$

- **Moda**: valor que aparece más veces

$$\max_j (n_j)$$

Ejemplo variable nominal y numérica

Edad	Sexo
23	M
25	M
18	H
37	M
45	H
62	H
43	M
40	H
60	M
54	H
28	H
18	H
54	M
29	H
42	M
26	M
32	M
41	M
37	M
36	H
53	H
21	M
24	H
21	H
45	M
64	H
22	M
61	M
37	M
66	M



Media y varianza de frecuencias estimadas

- Cálculo de cada frecuencia
  - para una categoría dada: m casos de n

    p=m/n
  - puede verse como asignar: vi=1 cada ejemplo en la categoría

    vi=0 en el resto

$$p = \frac{1}{n}\sum_{i=1}^{n} v_i$$

- Varianza de p:

$$Var(p) = \frac{1}{n}\sum_{i=1}^{n}(v_i - p)^2 = p(1-p)$$

$$\sigma_p = \sqrt{p(1-p)}$$

  - caso máxima varianza: p=0.5

Generalización de la muestra a la población

- **Los estadísticos** resumen (describen) toda la información contenida en una muestra (estadística descriptiva)
- Para generalizar las conclusiones, es deseable formular razonamientos sobre la población que genera la muestra
  - Paso de los estadísticos (yi) a los estimadores (Yi)
  - Uso de distribuciones teóricas de probabilidad para caracterizar los estimadores
  - Cuantificación de la probabilidad de los resultados (nunca se garantiza con certeza absoluta)
- Puede hacerse análisis contrario: deducción de propiedades de la muestra a partir de la población (interés teórico)

Distribuciones de probabilidad

- Modelo que representa la tendencia de un histograma con muchos datos y cajas pequeñas

- Función distribución de probabilidad de X: FX(x)

$$F_X(x) = P(X \le x); \qquad -\infty < x < \infty$$

- Función densidad de probabilidad de X: fX(x)

$$f_X(x) = \frac{dF_X(x)}{dx}; \qquad -\infty < x < \infty$$

$$F_X(x) = \int_{-\infty}^{x} f_X(x)dx; \quad P(a \le X \le b) = \int_a^b f_X(x)dx$$

Distribución Normal

- Curva de gran interés por explicar datos en muchas situaciones
    - Aplicada por primera vez como distribución por A. Quetelet (1830)

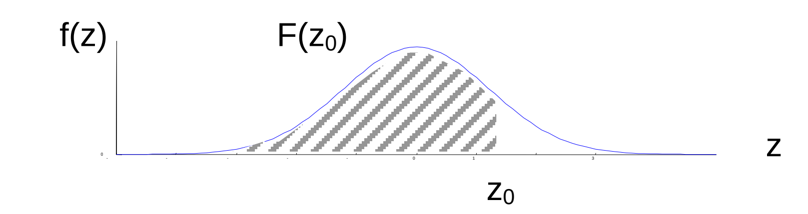
$$f(z) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}z^2\right]$$

- distribución simétrica: coincide media y mediana en 0
- se dispone del valor de la distribución de probabilidad: área bajo la curva de $f_Z(z)$ para cualquier valor:

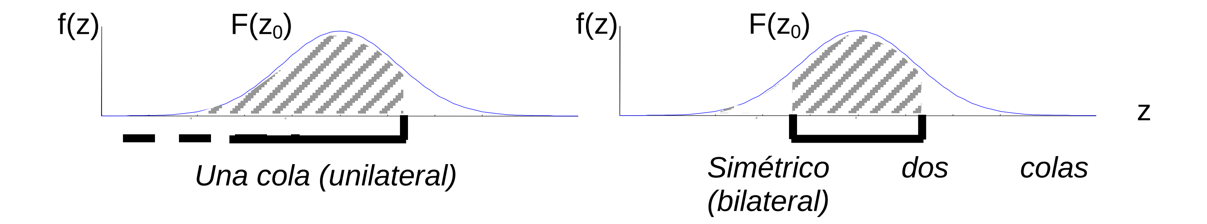
Tipificar o estandarizar variables: Se mide el desplazamiento respecto a la media en unidades de desviación típica:

$$z_i = \frac{y_i - \bar{y}}{\sigma_i}$$

z	$F_Z(z)$
-3	0,001349967
-2,5	0,00620968
-2	0,022750062
-1,5	0,066807229
-1	0,15865526
-0,5	0,308537533
0	0,5
0,5	0,691462467
1	0,84134474
1,5	0,933192771
2	0,977249938
2,5	0,99379032
3	0,998650033



Distribución Normal e Intervalos de Confianza



Una cola (unilateral)

Simétrico dos colas (bilateral)

- Ej.: se conocen parámetros de población con distribución normal: media: m= 115; desviación típica:s= 20
  - ¿casos inferiores a 70? z=(70-115)/20, F(z)=0,012
  - ¿casos superiores a 150? z=(150-115)/20, 1-F(z)=0,04
  - ¿en intervalo 90-130? F((130-115)/20)-F((90-115)/20)=0,667
  - ¿qué intervalos simétrico tienen el 80%, 95% de los casos (intervalos de confianza)? z=F-1(a/2); y=m±zs
    - 80%: z0.1=1,28; 115± z0.1*20=[89.3, 140.6]
    - 95%: z0.025=1,96; 115± z0.025*20=[75.8, 154.2]

Inferencia

- **Objetivo**: dado un estadísticos de una muestra sacada al azar, razonar acerca del verdadero parámetro de la población
- Se basa en la estimación de parámetros y contraste de hipótesis con cálculo de probabilidades
    - muestra aleatoria y representativa (estratificación)
    - elementos independientes
- Paso de la población a una muestra aleatoria
  - Dada una población con media y varianza:
  - Se toma una muestra aleatoria (n casos) de la población: yi, i=1,…,n
  - Cómo se distribuyen los estadísticos de la muestra? A su vez son VAs

Distribución de la media muestral

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$E(\bar{y}) = \frac{1}{n}\sum_{i=1}^{n} E(y_i) = \bar{Y}$$

$$Var(\bar{y}) = \frac{1}{n^2}\sum_{i=1}^{n} Var(y_i) = \frac{1}{n}Var(\bar{Y}); \quad \sigma_{\bar{y}} = \frac{1}{\sqrt{n}}\sigma_Y$$

- Qué distribución sigue? **Teorema del Límite Central:**

"Una muestra suficientemente grande de una población con distribución arbitraria tendrá estadístico media con distribución normal"

- **Consecuencia:** intervalo de confianza de la media a partir de dist. Normal

$$\bar{y} = \bar{Y} \pm z\frac{1}{\sqrt{n}}\sigma_Y$$

- Mayor "Normalidad": tamaño de las muestras, distribución pob. parecida a normal

Ejemplo límite central

- Población: 1000 individuos, 400 mujeres, 600 hombres

$$P = 0.4; \quad \sigma = \sqrt{P(1-P)} = 0.49$$

- Muestras de 10 individuos

$$p = \sum_{i=1}^{10} y_i;$$

$$E(p) = P = 0.4;$$

$$\sigma_p = \frac{1}{\sqrt{10}}\sqrt{P(1-P)} = 0.155$$

- Intervalo de confianza al 95% (con distribución normal):
  - Influye:
    - intervalo de confianza (z): "garantía" de no equivocarnos

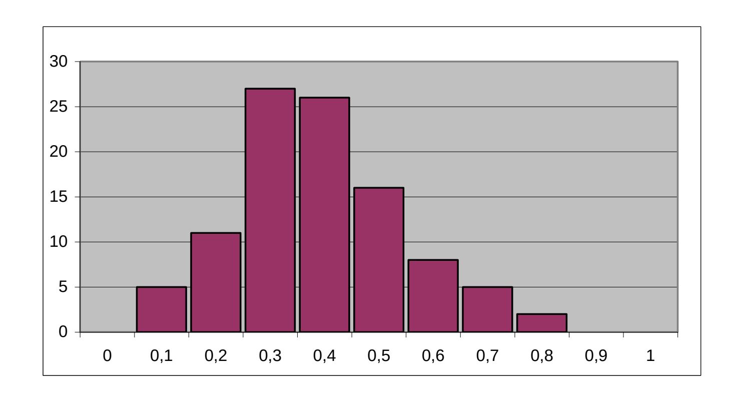
- tamaño de muestra (n)
- variabilidad de población (p)

$$p = \frac{1}{10}\sum_{i=1}^{10} y_i;$$

$$E(p) = P = 0.4;$$

$$\sigma_p = \frac{1}{\sqrt{10}}\sqrt{P(1-P)} = 0.155$$

$$P \pm 1.96\sigma_p = [0.1, 0.7]$$



- Si las muestras fueran de 50 individuos:

$$p = \frac{1}{50}\sum_{i=1}^{50} y_i;$$

$$\sigma_p = \frac{1}{\sqrt{50}}\sqrt{P(1-P)} = 0.069$$

$$P \pm 1.96\sigma_p = [0.26, 0.54]$$

Ejemplo de aplicación para decisión

- Para determinar el intervalo de confianza del estimador al 95% se aplica el argumento del muestreo “dado la vuelta”:

$$\bar{Y} \in \bar{y} \pm z_{\alpha/2}\sigma_{\bar{y}}$$

$$Ej: \quad P = p \pm 1.96\sigma_p$$

- Ejemplo: Un supermercado se plantea extender su horario a sábado por la tarde. Necesita un mínimo del 10% de sus clientes para cubrir costes. Con una muestra de 1500 personas se obtiene que hay un 8% de clientes interesados ¿Qué hacer?

$$P = p \pm 1.96\sigma_p = 0.08 \pm 1.96\sqrt{0.08*(1-0.08)/1500} =$$
$$[6.63\%, \quad 9.37\%]$$

- Con una confianza del 95% podemos decir que los clientes dispuestos a comprar el sábado por la tarde no contiene al deseado 10%.

Contrastes de hipótesis

- Contrastar es medir la probabilidad de que el estadístico obtenido en una muestra sea fruto del azar
- Formulación del modelo e hipótesis: se conoce la distribución del estadístico bajo condiciones hipótesis
- Hipótesis nula (**H0**): es lo que dudamos y queremos contrastar: Ej: ¿El porcentaje total es 10%?, la media de los ingresos es superior a 5?
    - Bajo H0, el estadístico sigue el modelo, y la diferencia observada es únicamente fruto del azar
- Hipótesis alternativas: alternativas que permiten rechazar la hipótesis nula: prob. distinta de 10%, media menor a 5, etc.
    - Rechazar hipótesis H0: hay evidencia para negar H0
    - No rechazable: no hay evidencia estadística para hacerlo (no implica demostrar su veracidad)

Contrastes con normal y varianza conocida

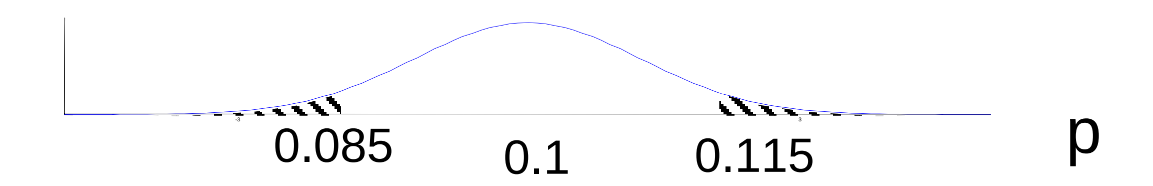
Contraste de dos colas (bilateral): deja la mitad a cada lado, a/2

- Ej: Hipótesis nula H0: P=10%

$$\sigma_p = \sqrt{0.1*(1-0.1)/1500}; \quad p \in [0.085 \quad 0.115]$$

- Hipótesis alternativa:

$$P \neq 10\%$$



- $z_{0.025}=1.96$

  $z_{0.05}=1.65$

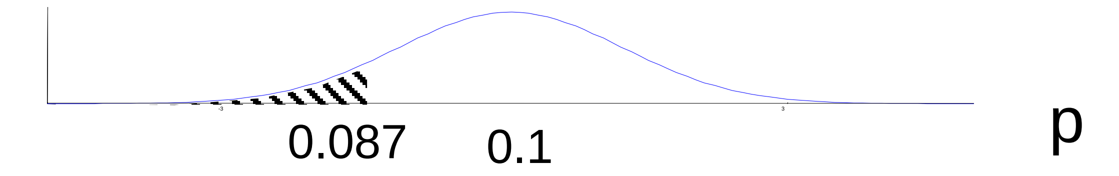
- Región crítica: -1,96<z<1.96

Contraste de una cola (unilateral): deja a un solo lado a

- Ej: Hipótesis nula H0:

$$p > P - 1.65\sigma_p = 0.087$$

- Hipótesis alternativa: P<10%



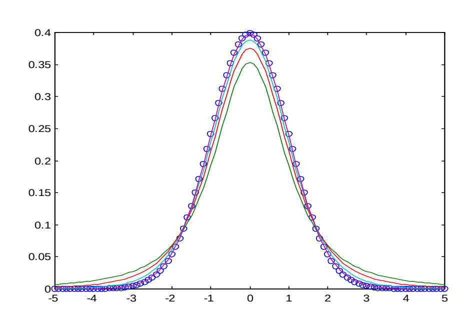
- Región crítica: z>1.65

Contraste con varianza estimada

- La variable (yi-y)/s no es exactamente la normal tipificada (s es estimada):
- Distribución **t-Student**: parámetro grados de libertad:n-1

- se ensanchan los intervalos de confianza (sólo si pocos datos)

	Student (N=9)	Student (N=50)	Student (N=100)	Normal
Prob[X>z]	z			z
0,10%	4,30	3,26	3,17	3,09
0,50%	3,25	2,68	2,63	2,58
1%	2,82	2,40	2,36	2,33
2,50%	2,26	2,01	1,98	1,96
5%	1,83	1,68	1,66	1,64
10%	1,38	1,30	1,29	1,28
20%	0,88	0,85	0,85	0,84



σ ,μ conocida	σ , μ conocida
estadístico	estadístico
$\frac{\bar{y}-\mu}{\sigma/\sqrt{n}} \rightarrow N(0,1)$	$\frac{\bar{y}-\mu}{\sigma/\sqrt{n}} \rightarrow t_{n-1}(0,1)$
Int. confianza	Int. confianza
$\bar{y} \pm z_{\alpha/2}\sigma/\sqrt{n}$	$\bar{y} \pm t_{\alpha/2,n-1}\sigma/\sqrt{n}$

Ejemplo de Intervalos con t-Student

Los valores del pH de una piscina en 10 determinaciones son: 6,8; 6,78; 6,77; 6,8; 6,78; 6,8; 6,82, 6,81; 6,8 y 6,79. Utilizando normal y t-Student, hallar:

- Intervalo de confianza 95% para media poblacional
- Intervalo de confianza 65% para media poblacional
- Contrastar hipótesis nula de que la media poblacional es 6,8 con niveles de significación a =0,05 y a=0,35

normal:

media 95%: [6,765, 6,825], media 65%: [6,781, 6,809]

t-Student:

media 95%: [6,761, 6,829], media 65%: [6,780, 6,801]

2.2. Técnicas de Evaluación de hipótesis

2.2.1. Análisis de relaciones entre atributos

El objetivo del análisis entre los atributos que definen los datos es ver el tipo de interrelación o dependencia que existe entre los valores de dichos atributos. Este análisis se lleva a cabo haciendo uso de los datos disponibles para tener “evidencia estadística” que permita validar o refutar hipótesis que pretendan explicar las relaciones.

La herramienta o técnica que permite llevar a cabo este tipo de análisis es el denominado tests de hipótesis, que se define de manera distinta en función del tipo de atributos con los que estemos trabajando. De esta manera en función del tipo de atributo tenemos:

- **Nominales-nominales**: En este caso los dos atributos toman valores de un conjunto de posibles valores (por ejemplo: Norte, Sur, Este y Oeste). La relación entre las variables se obtiene mediante las tablas de contingencia.

- **Nominales-numéricos**: En este caso uno de los atributos toma valores de un conjunto de posibles valores y otro toma valores numéricos. La relación entre los atributos se obtiene mediante la comparación de medias y el análisis de varianza.

- **Numéricos-numéricos**: En caso los dos atributos toman valores numéricos. La relación entre los dos atributos se obtiene mediante el análisis de regresión y covarianza.

Más adelante se contemplarán más casos de contrastes de hipótesis.

2.2.2. Relación entre variables nominales-nominales

El objetivo es analizar la interrelación (dependencia) entre los valores de variables nominales. En este caso la herramienta de análisis para dos variables es la denominada **tabla de contingencia**. En esta tabla se calcula la distribución de los casos (las frecuencias de aparición) para las distintas combinaciones de valores de las dos variables, como se oberva en la figura siguiente.

			Variable 2			totales 1
		valor 1	valor 2	...	valor p2	
Variable 1	valor 1	n_{11}	n_{12}	...	n_{1p2}	t1
	valor 2	n_{21}	n_{22}	...	n_{2p2}	t2
	...	...	...	...	...	...
	valor p1	n_{p11}	n_{p12}	...	n_{p1p2}	tp1
	totales 2	t'1	t'2	...	t'p2	t

Figura 1: *Tabla de contingencia.*

A partir de la tabla de contingencia podemos calcular las probabilidades marginales de los valores de la variable 1 como Pi=ti/t, que representa la probabilidad de que la variable 1 tome el valor i. Del mismo modo podemos calcular las probabilidades para la variable 2 como Pj=t'j/t.

A partir de las probabilidades marginales podemos calcular los casos "esperados", bajo la hipótesis a cuestionar de independencia entre variables. Para calcular el valor esperado se multiplica el número total de casos por la probabilidad de que la variable 1 tome el valor i y la variable 2 tome el valor j, es decir Eij=t(ti/t)(t'j/t)= tit'j/t. Obsérvese que únicamente bajo la hipótesis de independencia podemos calcular la probabilidad conjunta como un producto de probabilidades.

La técnica de análisis estadístico que se aplica para la relación entre dos variables nominales es el contraste Chi-2. Las características de este test son:

- Es aplicable en análisis bi-variable (normalmente clase vs atributo)
- Determina si es rechazable la hipótesis de que dos variables son independientes
    - Bajo hipótesis H0 se determinan los casos en el supuesto de variables independientes. Los valores esperados se determinan con probabilidades marginales de las categorías: Eij=tPi Pj (valores esperados)
    - El estadístico Chi-cuadrado mide la diferencia entre los valores esperados y los valores observados, por lo que su expresión es:

$$\chi^2 = \sum_{i=1}^{p1} \sum_{j=1}^{p2} (E_{ij} - O_{ij})^2 / E_{ij} \quad (1)$$

La expresión anterior, $\chi 2$, bajo las condiciones de H0 sigue una distribución conocida denominada distribución Chi-cuadrado, caracterizada por el parámetro *grados de libertad* que es el *(nº de filas-1)(nº de columnas –1)*. Cuando no se cumple la hipótesis H0 las variables son dependientes.

Por lo tanto se formula un test de hipótesis para determinar el valor de Chi-cuadrado para esa hipótesis. La distribución Chi-Cuadradado está tabulada:

probabilidad chi2 supera estadístico	valor estad						
grados de libertad	5	6	7	8	9	10	11
1	0,025	0,014	0,008	0,005	0,003	0,002	0,001
2	0,082	0,050	0,030	0,018	0,011	0,007	0,004
3	0,172	0,112	0,072	0,046	0,029	0,019	0,012
4	0,287	0,199	0,136	0,092	0,061	0,040	0,027
5	0,416	0,306	0,221	0,156	0,109	0,075	0,051
6	0,544	0,423	0,321	0,238	0,174	0,125	0,088
7	0,660	0,540	0,429	0,333	0,253	0,189	0,139

Y el test lo que calcula es la probabilidad de que la diferencia entre el valor observado y el valor esperado supere un cierto umbral.

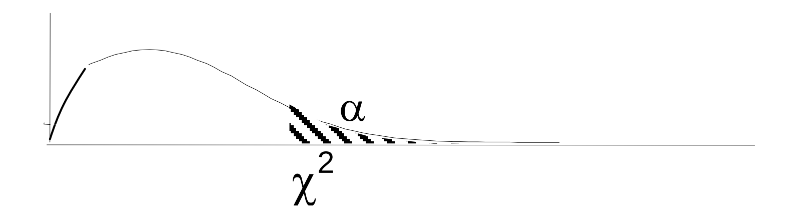


Figura 2: *Representación Gráfica del test Chi-Cuadrado.*

2.2.3. Relaciones numéricas-nominales

Las técnicas para establecer posibles relaciones entre dos variables una de ellas numérica y la otra nominal (o entre dos nominales si trabajamos con proporciones) se utiliza la técnica de la comparación de medias y proporciones. Esta técnica mide la relación entre variables numéricas y nominales, o nominales y nominales (proporciones), determinando si es rechazable la hipótesis de que las diferencias de medias o proporciones condicionadas a las etiquetas de la variable nominal son debidas al azar. Es decir que se calcula el impacto de la variable nominal sobre la continua.

Existen dos tipos de análisis según si tenemos dos medias o proporciones o un número mayor de dos. Si tenemos dos medias o proporciones se calcula la significatividad de la diferencia. Si tenemos más de dos valores distintos se realiza un análisis de varianza.

2.2.3.1. Comparación de dos medias

En este caso tenemos dos subpoblaciones, una para cada grupo, cada una con su media y varianza. Las hipótesis que podemos establecer son:

- H0: la diferencia de medias en la población es nula D=0
- Hipótesis alternativa A: las medias son distintas: D!=0
- Hipótesis alternativa B: la media de 1 es mayor que 2: D>0
- Hipótesis alternativa C: la media de 1 es mayor que 2: D<0

Como vemos, no hay una única posibilidad de hipótesis alternativa sino varias, con diferentes intervalos de rechazo en función de la información que tengamos a priori. Además, para la comparación de las variables numéricas de dos clases, las situaciones posibles que podemos encontrarnos dentro de la muestra total son:

- Muestras independientes: conjuntos distintos

- Muestras dependientes: es decir las muestras pertenecen al mismo conjunto, con dos variables a comparar en cada ejemplo

Cuando el número de muestras es muy elevado para cada grupo, las muestras siguen una distribución normal por lo que las hipótesis anteriormente expuestas se evalúan mediante los valores de una gaussiana estándar. De esta manera se calcularía la media de la diferencia y su varianza y se aplicaría al cálculo de probabilidades de una gaussiana estándar. En el caso de la hipótesis A se utilizarían las dos colas de la gaussiana y en el caso de la hipótesis B utilizaríamos una única cola, como se observa en la siguiente figura.

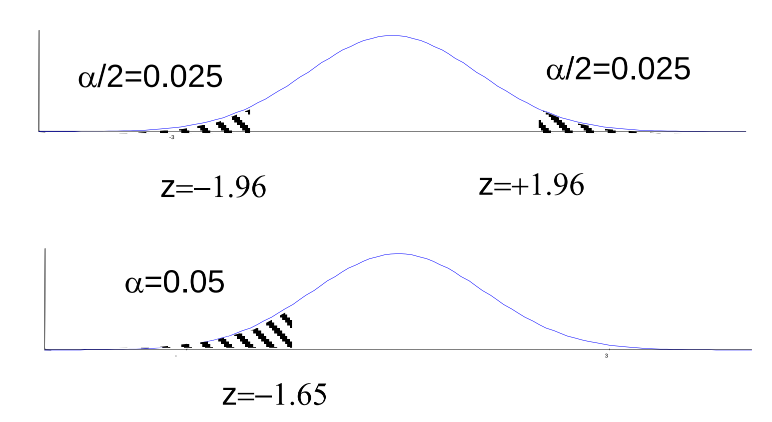


Figura 3: *Representación Gráfica de compración de dos medias medianteuna gaussiana.*

Cuando las muestras son pequeñas no es válida la hipótesis de normalidad de los estadísticos de medias y el test se realiza considerando una distribución t-Student:

$$\bar{y} \pm t_{\alpha/2,GL}\sigma \tag{2}$$

El proceso para el cálculo cuando las muestras son independientes (test no pareado) es:

- En cada muestra (tamaños n1, n2) obtenemos las medias y varianzas:

$$\bar{y}_1, \bar{y}_2, \sigma_{\bar{y}1}, \sigma_{\bar{y}2} \tag{3}$$

- Se calcula la diferencia:

$$d = \bar{y}_1 - \bar{y}_2 \tag{4}$$

- Varianza de la diferencia:

$$\sigma_d^{\ 2} = \frac{\sigma_{\bar{y}1}^{\ \ 2}}{n_1} + \frac{\sigma_{\bar{y}2}^{\ \ 2}}{n_2} \qquad (5)$$

- Los grados de libertad de la t-Student se evalúan según la varianza:
  - Distinta varianza (*heteroscedasticidad*): gl=min(n1, n2)
  - Misma varianza (*homoscedasticidad*): gl=n1+n2-2

El proceso de cálculo cuando las muestras dependientes (test pareado), se fundamenta en que se dispone de la diferencia en cada uno de los ejemplos y no en que tenemos dos variables (ejemplo: cambio en el tiempo de una variable para todos los ejemplos d1, d2, ..., dn): di=d1i-d2i. En este caso todo es equivalente al caso anterior pero lo cálculos son:

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i; \quad \sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(d_i - d)^2; \quad \sigma_{\bar{d}} = \frac{1}{\sqrt{n}}\sigma \qquad (6)$$

2.2.3.2. Análisis de la varianza

Esta técnica también mide la relación entre variables numéricas y nominales, pero en este caso se descompone la variabilidad del resultado en varios componentes:

- Efectos de factores representados por otras variables
- Efectos de error experimental

La técnica del análisis de la varianza simple (ANOVA) considera un solo factor con varios niveles nominales. Para cada nivel se tiene una serie de observaciones y el modelo: Yij=ui+uij, representa ruido con la misma varianza por nivel, donde i varía entre 1 y el número de niveles (variable nominal) y j varía entre 1 y el número de datos por nivel. Además de esta técnica existe la técnica MANOVA que es un modelo multifactorial de la varianza. En este modelo se definen I niveles, cada uno de ellos representado por un conjunto de muestras, como se puede observar en la siguiente figura, y donde cada nivel está represntado por una media y una varianza.

	Factor B	1	2	…	r
Factor A					
		X_{111}	X_{121}	…	X_{1r1}
		X_{112}	X_{122}	…	X_{1r2}
1		…	…	…	…
		X_{11n11}	X_{12n12}	…	X_{1rn1r}
		X_{211}	X_{221}	…	X_{2r1}
		X_{212}	X_{222}	…	X_{2r2}
2		…	…	…	…
		X_{21n21}	X_{22n22}	…	X_{2rn2r}
…		…	…	…	
		X_{t11}	X_{t21}	…	X_{tr1}
		X_{t12}	X_{t22}	…	X_{tr2}
t		…	…	…	…
		X_{t1nt1}	X_{t2nt2}	…	X_{trntr}

Figura 4: *Niveles de la técnica MANOVA.*

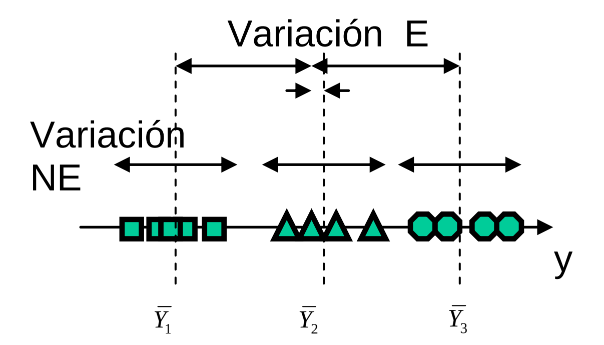


Figura 5: *Represntación Gráfica de los Niveles de la técnica MANOVA.*

El análisis MANOVA evalúa las siguientes variables:

- Número total de elementos:

$$n = \sum_{i=1}^{I} n_i \tag{7}$$

- Media por nivel:

$$\bar{Y}_i = \frac{1}{n_i} \sum_{i=1}^{I} Y_{ij} \tag{8}$$

- Media total:

$$\overline{Y} = \frac{1}{n}\sum_{i=i}^{I}\sum_{j=1}^{n_i} Y_{ij} \tag{9}$$

- Relación entre "cuadrados":

$$\sum_{i=i}^{I}\sum_{j=1}^{n_i}(Y_{ij} - \overline{Y})^2 = \sum_{i=i}^{I}\sum_{j=1}^{n_i}(Y_{ij} - \overline{Y}_i)^2 + \sum_{i=1}^{I} n_i(\overline{Y}_i - \overline{Y}) \tag{10}$$

Y realiza una estimación de varianzas de la siguiente manera

- Varianza inter-grupo (*between*) (I-1 grados de libertad):

$$S_b = \frac{1}{I-1}\sum_{i=1}^{I} n_i(\overline{Y}_i - \overline{Y})^2 \tag{11}$$

- Varianza intra-grupo (*within*) (n-I grados de libertad):

$$S_w = \frac{1}{n-I}\sum_{i=i}^{I}\sum_{j=1}^{n_i}(Y_{ij} - \overline{Y}_i)^2 \tag{12}$$

- Varianza total (n-1 grados de libertad):

$$S = \frac{1}{n-1}\sum_{i=i}^{I}\sum_{j=1}^{n_i}(Y_{ij} - \overline{Y})^2 \tag{13}$$

La hipótesis que planteamos o la pregunta que queremos responder es: ¿Es significativamente mayor que la unidad la relación entre la varianza intergrupo e intragrupo, f=Sb/Sw?. Por lo tanto debemos realizar un contraste de hipótesis de cociente de varianzas maestrales, que sigue una distribución **F de Fisher-Snedecor**: F(x, I-1,n-I), como se ve en la figura siguiente.

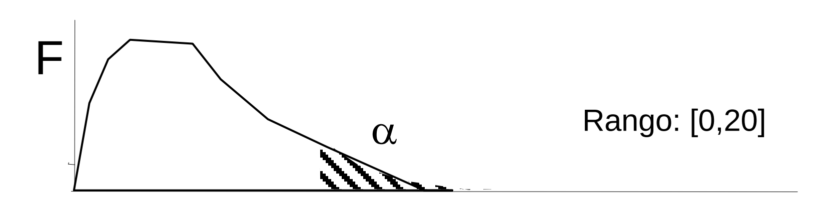


Figura 6: *Representación de la F-Fisher-Snedecor.*

Este test permite rechazar o no la hipótesis de que el cociente entre varianzas estimadas se deba al azar. Por lo tanto

2.2.4. Relaciones numéricas-numéricas:

2.2.4.1. Regresión lineal

La regresión lineal permite identificar relaciones entre variables numéricas y construir modelos de regresión: 1 variable salida y múltiples entradas numéricas. Se consideran relaciones de una variable de salida (dependiente) con múltiples variables de entrada (independientes). Este problema se puede representar de la siguiente manera:

Dada la muestra de datos: $\{(\vec{X}_1, y_1), (\vec{X}_2, y_2), \ldots, (\vec{X}_n, y_n)\}$ donde $\vec{X}: \quad \text{vectores con I dimensiones}$, se busca estimar una función que mejor "explique" los datos:

$$\begin{aligned} g(.): \quad & R^I \longrightarrow R \\ & \vec{X} \longrightarrow \hat{y} = g(\vec{X}) \end{aligned} \qquad (14)$$

El procedimiento de resolución para estimar dicha función es el procedimiento de mínimos cuadrados que estima el vector de coeficientes que minimiza el error:

$$\begin{aligned} & \hat{y}_i = g_i(\vec{X}) = a_0 + \sum_{p=1}^{I} a_p x_p = (\vec{A}^t) * \vec{X} \\ & (\vec{A}) = [a_0 \quad a_1 \quad \cdots \quad a_I]^t; \quad \vec{X} = [1 \quad x_1 \quad \cdots \quad x_I]^t \end{aligned} \qquad (15)$$

El objetivo es que dadas N muestras, el procedimiento debe determinar coeficientes que minimicen el error de predicción global

$$\varepsilon = \sum_{j=1}^{n} [g(\vec{X}_j) - y_j]^2 \qquad (16)$$

Este es un problema clásico de minimización de función cuadrática cuya solución es única. La formulación genérica matricial del problema se puede expresar como:

$$\vec{y} = \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix}; \quad \hat{g} = \begin{bmatrix} \hat{y}^1 \\ \vdots \\ \hat{y}^N \end{bmatrix} = \begin{bmatrix} g(\vec{X}^1) \\ \vdots \\ g(\vec{X}^N) \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & \cdots & x_I^1 \\ 1 & x_1^2 & \cdots & x_I^2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^N & \cdots & x_I^N \end{bmatrix} \vec{A} = H * \vec{A} \qquad (17)$$

Por lo que la solución de mínimos cuadrados es: $\vec{A} = \left[H^t H\right]^{-1} H^t \vec{y}$

2.2.5. Evaluación del modelo de regresión

La evaluación del modelo realiza el análisis de validez del modelo asumido, es decir se van a calcular una serie de medidas de “parecido” entre la variable de salida estimada mediante la función y los valores de la variable de salida real, ide esta manera analizaremos la nfluencia de las variables de entrada en el cálculo de la variable de salida (si existe o no una relación lineal entre las variables de entrada que permita determinar la variable de salida). Estas medidas son: el Factor de Correlación (que muestra si existe la relación lineal), el error de predicción (diferencia entre la predicha y la real) y el error en coeficientes.

2.2.5.1. Medidas de Calidad

El factor de correlación se evalúa como:

$$\begin{aligned} &Corr(\hat{y},\, y) = \frac{1}{\sqrt{S_{\hat{y}} S_y}} \sum_{j=1}^{n} (\hat{y}_j - \bar{\hat{y}})(y_j - \bar{y}) = \frac{Cov(\hat{y},\, y)}{\sqrt{Var(\hat{y}) Var(y)}} \\ &S_{\hat{y}} = \sum_{j=1}^{n} \left(\hat{y}_j - \bar{\hat{y}}\right)^2; \quad S_y = \sum_{j=1}^{n} \left(y_j - \bar{y}\right)^2; \\ &\bar{\hat{y}} = \frac{1}{N} \sum_{j=1}^{N} \hat{y}_j, \qquad \bar{y} = \frac{1}{N} \sum_{j=1}^{N} y_j \end{aligned} \tag{18}$$

En general, se puede hacer factores de correlación entre cualquier par de variables numéricas: indica el grado de relación lineal existente. Para ello se calcula la matriz de covarianzas (o la de correlaciones que es la misma pero normalizada) de la siguiente manera:

$$\hat{C}\infty_{\vec{X}} = \frac{1}{n} \sum_{i=1}^{n} \left(\vec{X}_i - \hat{\vec{\mu}}\right)^t = \begin{bmatrix} \mathrm{var}(x_1) & \mathrm{cov}(x_1, x_2) & \cdots & \mathrm{cov}(x_1, x_2) \\ \mathrm{cov}(x_1, x_2) & \mathrm{var}(x_2) & & \vdots \\ \vdots & & \ddots & \\ \mathrm{cov}(x_1, x_I) & \cdots & & \mathrm{var}(x_I) \end{bmatrix} \tag{19}$$

donde $\hat{\vec{\mu}} = \frac{1}{n} \sum_{i=1}^{n} \vec{X}_i$

El error de predicción se evalúa como: $Error = \sum_{j=1}^{n} \left(\hat{y}_j - y_j\right)^2 = \sum_{j=1}^{n} \left(\varepsilon_j\right)^2$ bajo la hipótesis de que los datos y_i tengan la misma varianza *sy*, sean independientes, y que el modelo lineal sea adecuado el error puede calcularse como: $Error \approx (n-1)\sigma_y^2$.

El error en coeficientes se evalúa a partir de la expresión que permite encontrar los coeficientes $\vec{A} = [H^t H]^{-1} H^t \vec{y}; \quad \vec{\varepsilon}_A = [H^t H]^{-1} H^t \vec{\varepsilon}_y$. La relación entre los errores en predicción y en coeficientes estimados se evalúa:

$$C_A = \begin{bmatrix} \sigma^2_{A_0} & \cdots & \cdots & \\ \vdots & \sigma^2_{A_1} & & \vdots \\ \vdots & & \ddots & \vdots \\ & \cdots & \cdots & \sigma^2_{A_I} \end{bmatrix} = \sigma^2_y [H^t H]^{-1} \qquad (20)$$

Por lo que el error en los coeficientes depende de el error en y, sy2 y el recorrido de datos X, es decir la matriz H.

2.2.5.2. Test de Hipótesis sobre modelo de regresión

Estos valores permiten analizar la “calidad” del modelo mediante los test de hipótesis: hipótesis de significatividad de parámetros (gaussiana o t-Student) y la hipótesis de ausencia de relación (F de Fisher-Snedecor).

Para evaluar la significatividad de parámetros, partimos de varianzas de parámetros {s2A1,…s2AF} y los propios valores estimados, y nos preguntamos si son significativos los parámetros: $\frac{A_1}{\sigma_{A_1}},...,\frac{A_F}{\sigma_{A_F}}?$. Este test puede resolverse mediante una gaussiana estándar si tenemos gran cantidad de datos, o bien, si hay pocos datos: en vez de estadística normal, una t-Student con n-F-1 grados de libertad. También podemos extender el modelo y analizarlo: ej: dependencia cuadrática, ver si son significativos nuevos términos

Para analizar la validez del modelo debemos realizar un análisis de la varianza que permite rechazar o no la hipótesis de que no existe relación entre variables (relación debida al azar, correlación nula). Para ello a partir del valor:

$$\sum_{j=1}^{N}\left(y_j - \bar{y}\right)^2 = \sum_{j=1}^{N}\left(\hat{y}_j - \bar{y}\right)^2 + \sum_{j=1}^{N}\left(\hat{y}_j - y\right)^2 \qquad (21)$$

calculamos el estadístico : $F = \frac{SE/I}{SR/(n-I-1)}$ que sigue una distribución: F de Snedecor: F(n1, n2), donde los grados de libertad son: I, n-I-1

2.3. Ejemplos de aplicación de técnicas de evaluación de hipótesis

Con un objetivo meramente ilustrativo, en esta sección se sugieren algunas aplicaciones de las técnicas de contraste de hipótesis y minería de datos presentadas en otras secciones. Son ejemplos que se relacionarían con el objetivo final de este proyecto de analizar y describir relaciones de interés y modelos subyacentes en datos del dominio del tráfico aéreo. Hay que tener en cuenta, que son ejemplos sugeridos que quedarían sujetos a su validación mediante la generación de los datos apropiados, sujeto a una metodología apropiada de preparación, interpretación y validación.

2.3.1. Ejemplos de Validación de Hipótesis

Para ilustrar la técnica de contraste de hipótesis para independencia entre variables de tipo nominal, supongamos que partimos de los datos de la tabla siguiente:

tipo avión/ retardo	Ligero	Mediano	Pesado	**total**
nulo	75	323	15	**413**
medio	32	405	6	**443**
alto	8	45	7	**60**
muy alto	2	15	1	**18**
total	**117**	**788**	**29**	**934**

En esta tabla se representan dos variables nominales: retardo y tipo de avión. La variable retardo puede tomar 4 valores: nulo, medio, alto y muy alto. La variable tipo de avión puede tomar 3 valores: Ligero, Mediano y Pesado. En la tabla aparecen el número de aviones de cada tipo en función del retardo que sufren. Es decir, aparece la distribución observada para el número de aviones de cada tipo que sufre una determinada categoría de retardo.

Si en la tabla anterior consideramos únicamente los valores totales de las variables tipo de avión y retardo, podemos calcular la probabilidad de cada categoría dividiendo del total marginal por el número total de casos. Además, en el caso hipotético de que fueran las dos variables independientes, la probabilidad conjunta de cada casilla sería el producto de estas probabilidades, y multiplicada por el número total de casos tendríamos el valor esperado en cada casilla. Eij=t(ti/t)(t'j/t)

Así, por ejemplo, para la combinación avión ligero y retardo nulo, tendríamos:

$$N_{tipo=ligero,\ retardo=nulo} = \frac{413}{934}\frac{117}{934}934 = 51.74$$

Repitiendo el mismo proceso para el resto de casillas, tenemos:

tipo avión/ retardo	Ligero	Mediano	Pesado	total
nulo	**51,74**	**348,44**	**12,82**	413
medio	**55,49**	**373,75**	**13,75**	443
alto	**7,52**	**50,62**	**1,86**	60
muy alto	**2,25**	**15,19**	**0,56**	18
total	117	788	29	934

Por lo tanto a partir de dichos datos podemos plantearnos la hipótesis nula H0: las variables retardo y categoría son independientes. Calculando el estadístico que acumula las desviaciones cuadráticas divididas por los valores esperados tenemos:

$$\chi^2 = \sum_{i=1}^{p1} \sum_{j=1}^{p2} (E_{ij} - O_{ij})^2 / E_{ij} \qquad (22)$$

y evaluamos la probabilidad del estadístico mediante la función Chi-cuadrado. Tomando 3x2 grados de libertad, tenemos que el valor de corte al 95% para rechazar sería de 12.59 (ver siguiente Figura).

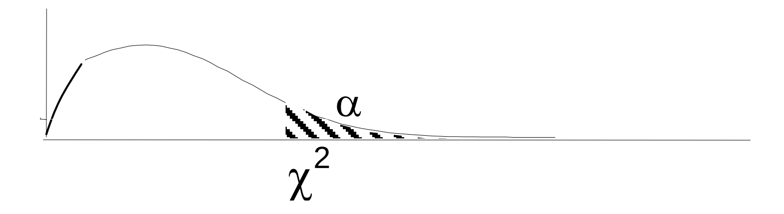
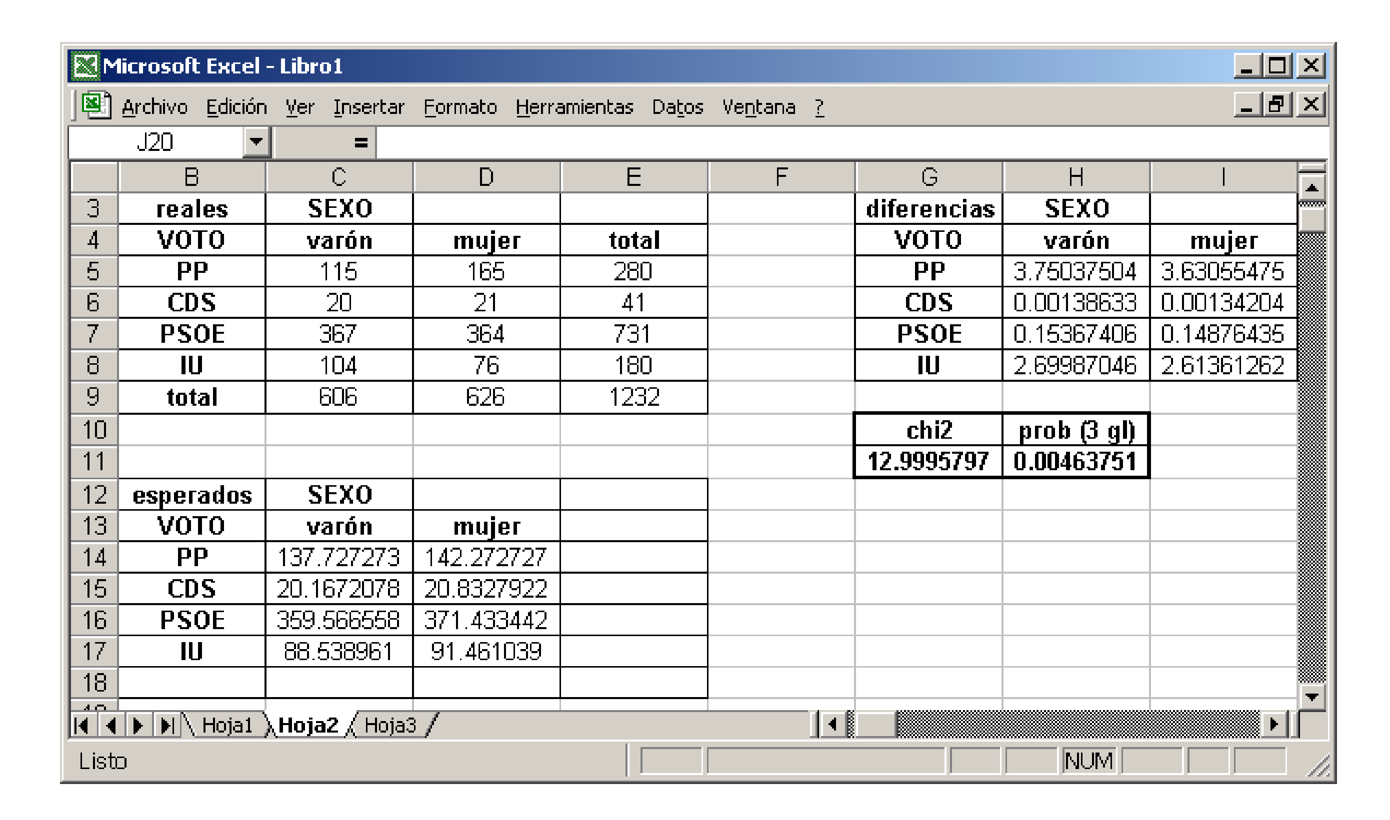


Figura 7: *Test Chi-Cuadrado.*

Sin embargo, con los valores observados, tenemos que la desviación es 44,91, que para una distribución Chi-cuadrado de 6 grados de libertad tiene una probabilidad de aparecer de 4,87e-8, lo que nos permite rechazar con mucha evidencia la hipótesis de independencia y concluir una clara dependencia entre las variables.

El ejemplo siguiente aplica la misma técnica para determinar la interdependencia entre la intención de voto y el sexo en una población dada:



Microsoft Excel - Libro1

Archivo Edición Ver Insertar Formato Herramientas Datos Ventana ?

J20 =

	B	C	D	E	F	G	H	I
3	reales	SEXO				diferencias	SEXO	
4	VOTO	varón	mujer	total		VOTO	varón	mujer
5	PP	115	165	280		PP	3.75037504	3.63055475
6	CDS	20	21	41		CDS	0.00138633	0.00134204
7	PSOE	367	364	731		PSOE	0.15367406	0.14876435
8	IU	104	76	180		IU	2.69987046	2.61361262
9	total	606	626	1232				
10						chi2	prob (3 gl)	
11						12.9995797	0.00463751	
12	esperados	SEXO						
13	VOTO	varón	mujer					
14	PP	137.727273	142.272727					
15	CDS	20.1672078	20.8327922					
16	PSOE	359.566558	371.433442					
17	IU	88.538961	91.461039					
18								

Hoja1 Hoja2 Hoja3

Listo NUM

Relaciones numéricas-numéricas: regresión lineal

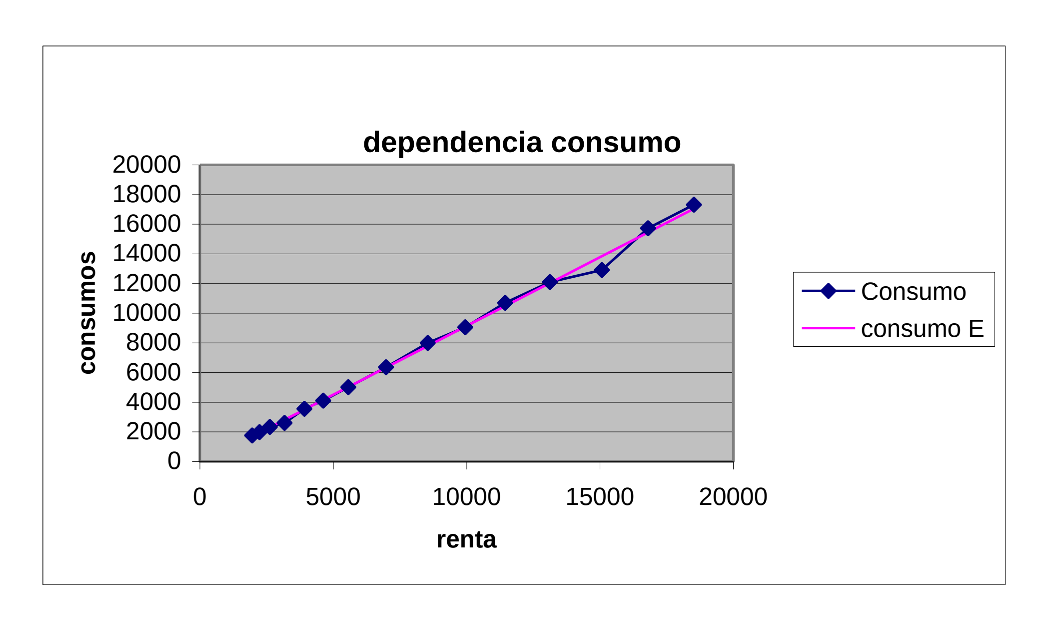
- Permite identificar relaciones entre variables numéricas y construir modelos de regresión: 1 variable salida y múltiples entradas numéricas
- Se consideran relaciones de una variable de salida (dependiente) con múltiples variables de entrada (independientes)

Ejemplo: regresión lineal de 1 variable

Año	Renta	Consumo	consumo E
1970	1959,75	1751,87	1683,473374
1971	2239,09	1986,35	1942,43325
1972	2623,84	2327,9	2299,11261
1973	3176,06	2600,1	2811,043671
1974	3921,6	3550,7	3502,190468
1975	4624,7	4101,7	4153,993607
1976	5566,02	5012,6	5026,63666
1977	6977,84	6360,2	6335,452914
1978	8542,51	7990,13	7785,967518
1979	9949,9	9053,5	9090,676976
1980	11447,5	10695,4	10479,01488
1981	13123,04	12093,8	12032,31062
1982	15069,5	12906,27	13836,76054
1983	16801,6	15720,1	15442,48976
1984	18523,5	17309,7	17038,76316

Estimación Lineal	
a1	a0
0,927041871	-133,296932

$$ConsumoE = a0 + a1 * Renta$$

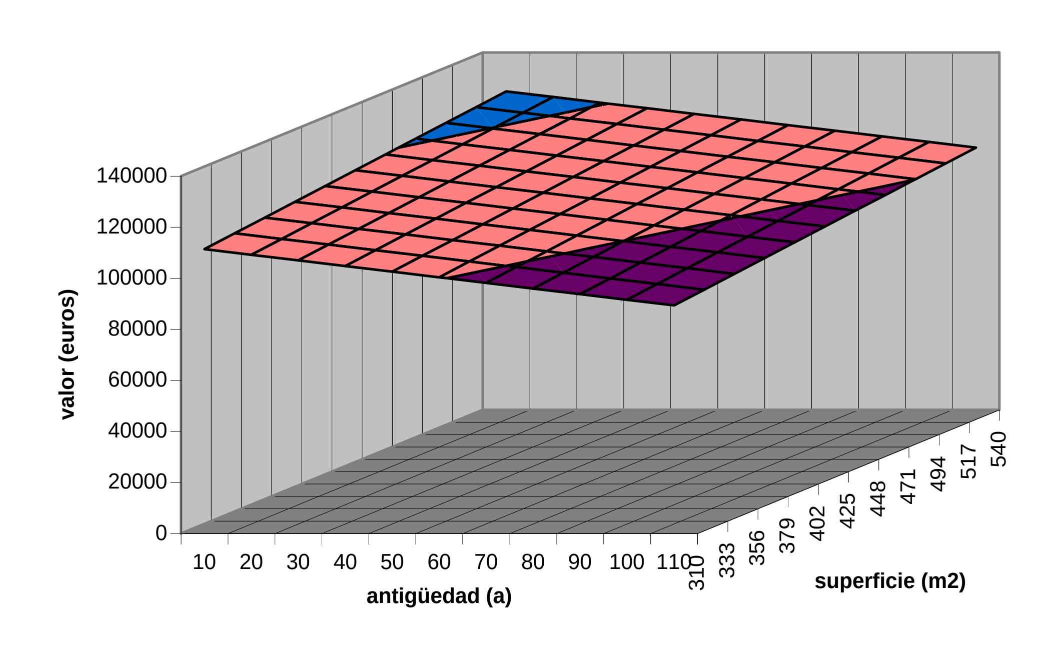


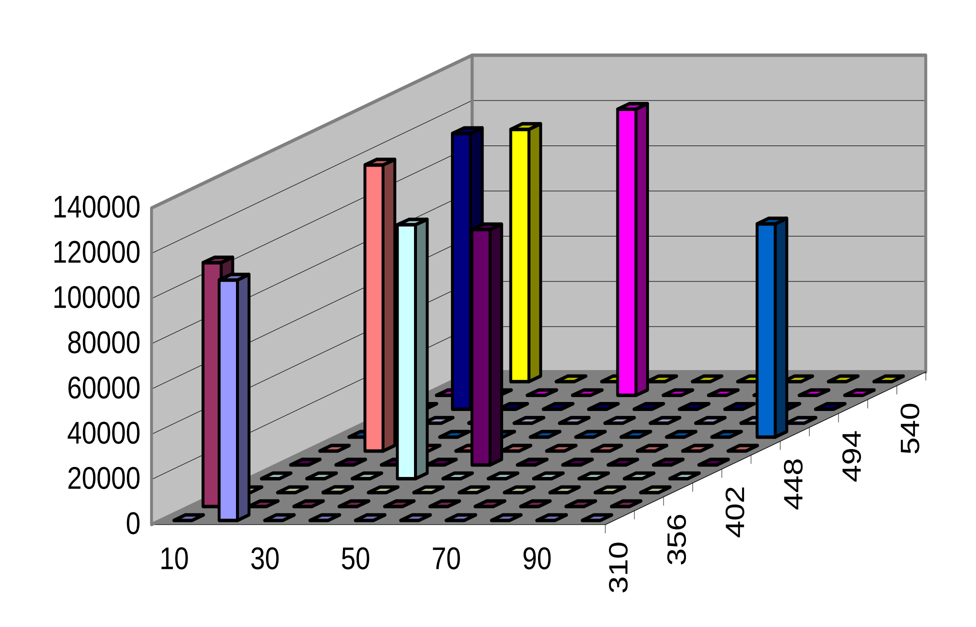
Ejemplo: regresión lineal de 2 variables

x1 Superficie	x2 Antigüedad	y Valor	Valor predicho
310	20	106.287 Euros	109.180 Euros
333	12	107.784 Euros	112.283 Euros
356	33	113.024 Euros	108.993 Euros
379	43	112.275 Euros	108.128 Euros
402	53	104.042 Euros	107.262 Euros
425	23	126.497 Euros	115.215 Euros
448	99	94.311 Euros	99.800 Euros
471	34	106.961 Euros	115.469 Euros
494	23	122.006 Euros	119.233 Euros
517	55	126.497 Euros	113.518 Euros
540	22	111.527 Euros	122.132 Euros

Estimación Lineal		
a2	a1	a0
-220,444829	58,2271936	95538,7217

$$\text{Valor} = \text{a0} + \text{a1} * \text{Superficie} + \text{a2} * \text{Antigüedad}$$





Evaluación del modelo de regresión

Análisis de validez del modelo asumido:

- Medidas de “parecido” entre variable de salida estimada y real, influencia de variables de entrada
  - Factor de Correlación
  - Error de predicción
  - Error en coeficientes

- Análisis de “calidad” del modelo
  - Hipótesis de significatividad de parámetros: t-Student
  - Hipótesis de ausencia de relación: F de Fisher-Snedecor

Factor de correlación

Factor de correlación entre datos y predicciones:

$$Corr(\hat{y},\, y) = \frac{1}{\sqrt{S_{\hat{y}} S_y}} \sum_{j=1}^{n} (\hat{y}_j - \bar{\hat{y}})(y_j - \bar{y}) = \frac{Cov(\hat{y}, y)}{\sqrt{Var(\hat{y})Var(y)}}$$

$$S_{\hat{y}} = \sum_{j=1}^{n} \left(\hat{y}_j - \bar{\hat{y}}\right)^2; \quad S_y = \sum_{j=1}^{n} \left(y_j - \bar{y}\right)^2;$$

$$\bar{\hat{y}} = \frac{1}{N} \sum_{j=1}^{N} \hat{y}_j, \qquad \bar{y} = \frac{1}{N} \sum_{j=1}^{N} y_j$$

En general, se puede hacer factores de correlación entre cualquier par de variables numéricas: indica el grado de relación lineal existente

Matriz de Covarianza

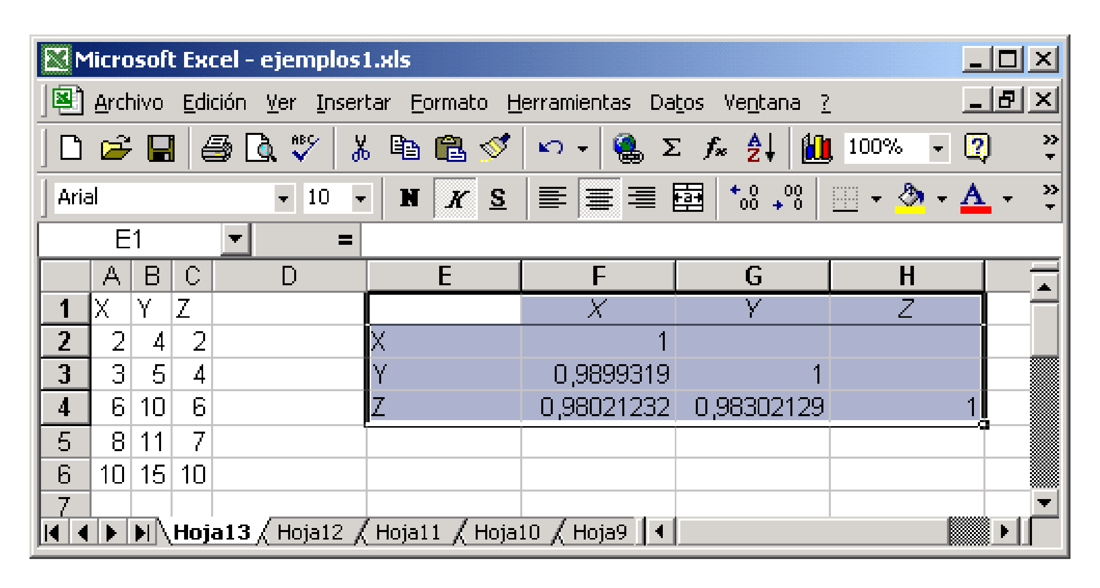
Muestra de vectores aleatorios:

- Matriz de covarianzas:

$$\hat{\vec{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\vec{X}_i$$

$$\hat{C}_{\vec{X}} = \frac{1}{n}\sum_{i=1}^{n}(\vec{X}_i - \hat{\vec{\mu}})(\vec{X}_i - \hat{\vec{\mu}})^t = \begin{bmatrix} \mathrm{var}(x_1) & \mathrm{cov}(x_1,x_2) & \cdots & \mathrm{cov}(x_1,x_2) \\ \mathrm{cov}(x_1,x_2) & \mathrm{var}(x_1) & & \\ \vdots & & \ddots & \vdots \\ \mathrm{cov}(x_1,x_I) & \cdots & & \mathrm{var}(x_I) \end{bmatrix}$$

- La matriz de correlaciones es similar, normalizada



Error de Predicción

$$Error = \sum_{j=1}^{n}\left(\hat{y}_j - y_j\right)^2 = \sum_{j=1}^{n}\left(\varepsilon_j\right)^2$$

bajo la hipótesis de que los datos yi tengan la misma varianza sy, sean independientes, y que el modelo lineal sea adecuado:

$$Error \approx (n-1)\sigma_y^2$$

Error en coeficientes?

$$\vec{A} = [H^t H]^{-1} H^t \vec{y}; \quad \vec{\varepsilon}_A = [H^t H]^{-1} H^t \vec{\varepsilon}_y$$

relación errores en predicción y en coeficientes estimados:

$$C_A = \begin{bmatrix} \sigma_{A_0}^2 & \cdots & \cdots & \\ \vdots & \sigma_{A_1}^2 & & \vdots \\ \vdots & & \ddots & \vdots \\ & \cdots & \cdots & \sigma_{A_I}^2 \end{bmatrix} = \sigma_y^2 [H^t H]^{-1}$$

El error en los coeficientes depende de

- error en y, sy2
- recorrido de datos X: matriz H

x	y
5,33	8,15
5,65	7,84
7,27	9,33
8,05	10,07
8,66	11,60
8,80	11,48
8,89	11,89
8,98	11,12
9,35	12,01
9,82	12,01

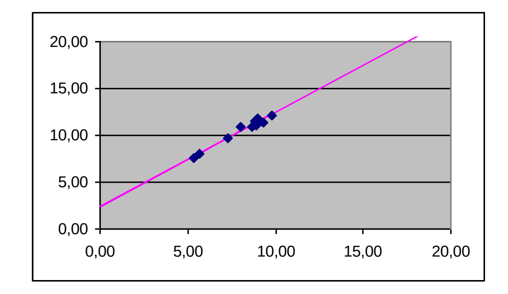
Rango: [5,10]

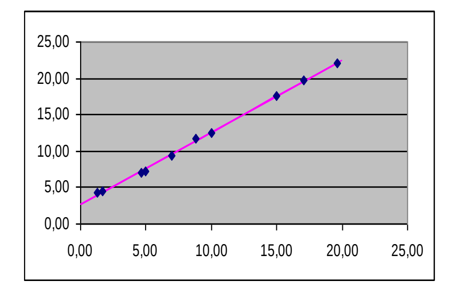
σ_y=1

σ_{A0}=0.6

σ_{A1}=0.07

x	y
1,32	3,67
1,68	4,66
4,69	7,57
4,99	7,48
6,98	9,66
8,80	11,51
10,01	12,02
15,01	17,47
17,10	19,82
19,67	21,94

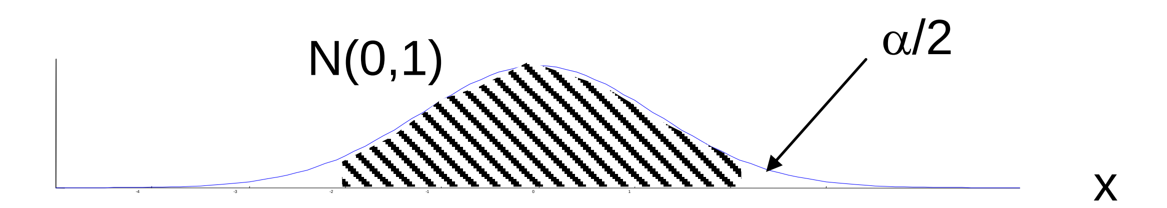




Significatividad de parámetros

- Dadas las varianzas de parámetros {s2A1,…s2AF} y los propios valores estimados, son significativos los parámetros?

$$\frac{A_1}{\sigma_{A_1}},\ldots,\frac{A_F}{\sigma_{A_F}}\ ?$$



- Si hay pocos datos: en vez de estadística normal, t-Student con n-F-1 grados de libertad
    - Posibilidad de extender el modelo y analizarlo: ej: dependencia cuadrática, ver si son significativos nuevos términos

Validez del modelo: análisis de varianza

- Permite rechazar o no la hipótesis de que no existe relación entre variables (relación debida al azar, correlación nula)

$$\sum_{j=1}^{N}\left(y_j-\bar{y}\right)^2=\sum_{j=1}^{N}\left(\hat{y}_j-\bar{y}\right)^2+\sum_{j=1}^{N}\left(\hat{y}_j-y\right)^2$$

- Estadístico

$$F=\frac{SE/I}{SR/(n-I-1)}$$

distribución: F de Snedecor: F(n1, n2)

grados de libertad: I, n-I-1

2.4. Técnicas clásicas de clasificación y predicción

- Modelado de datos con atributos numéricos para su aplicación a Clasificación. Generalización
    - Datos representados como vectores de atributos numéricos: patrones
- Clases: {C1, ..., CM}
- Muestras:E=

$$\{\vec{X}_1^{(1)},...,\vec{X}_{n_1}^{(1)},\vec{X}_1^{(2)},...,\vec{X}_{n_2}^{(2)},...,\vec{X}_1^{(M)}...,\vec{X}_{n_M}^{(M)}\}$$

    – Tamaño:

$$n=\sum_{j=1}^{M}n_j$$

- Para cada clase, Ci, hay ni patrones, cada uno con I atributos: para cada clase Ci:

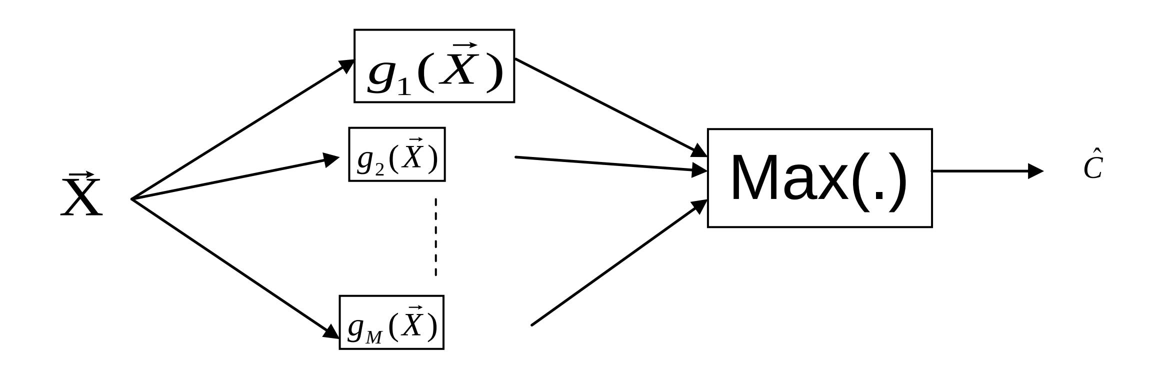
$$\{\vec{X}_1^{(i)},...,\vec{X}_{n_i}^{(i)}\}$$

$$\vec{X}_j^{(i)}=\begin{bmatrix}x_{1j}^{(i)}\\ \vdots\\ x_{Ij}^{(i)}\end{bmatrix};\quad j=1,...,n_i$$

$$g(.):\quad R^I\longrightarrow C=\{C_1,...,C_M\}$$

$$\vec{X}\longrightarrow\hat{\mathrm{C}}=\mathrm{g}(\vec{X})$$

- Función discriminante de cada clase:

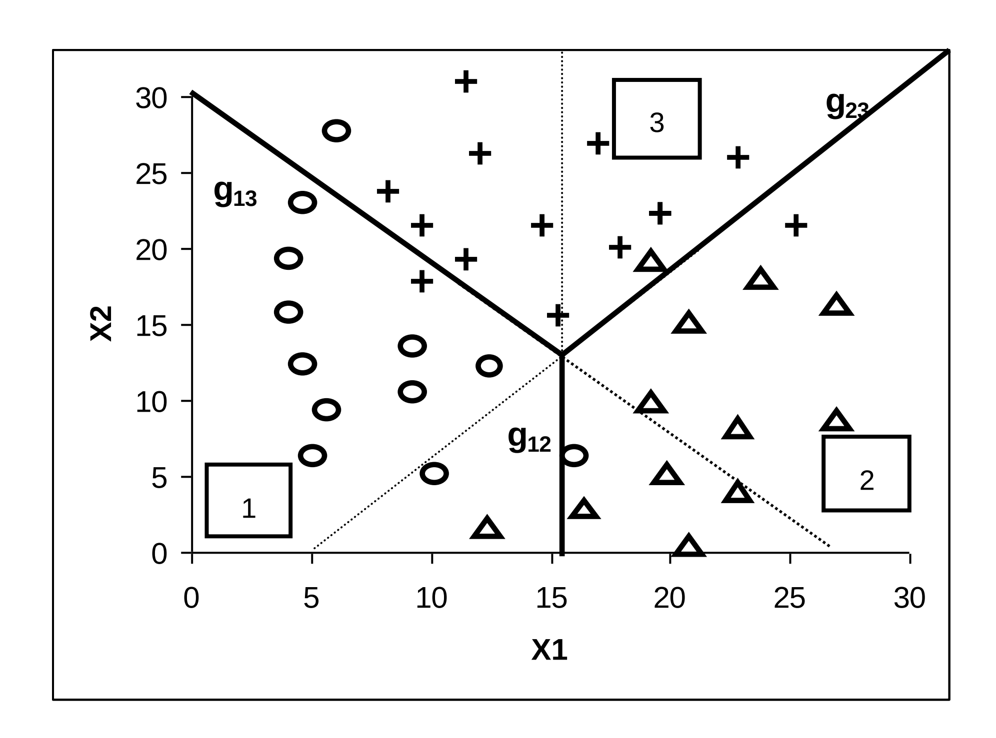


- Propiedad deseable para el diseño de gi(.): sobre el conjunto de entrenamiento, cada patrón de la clase **Ci** tiene un valor máximo con el discriminante gi(.):

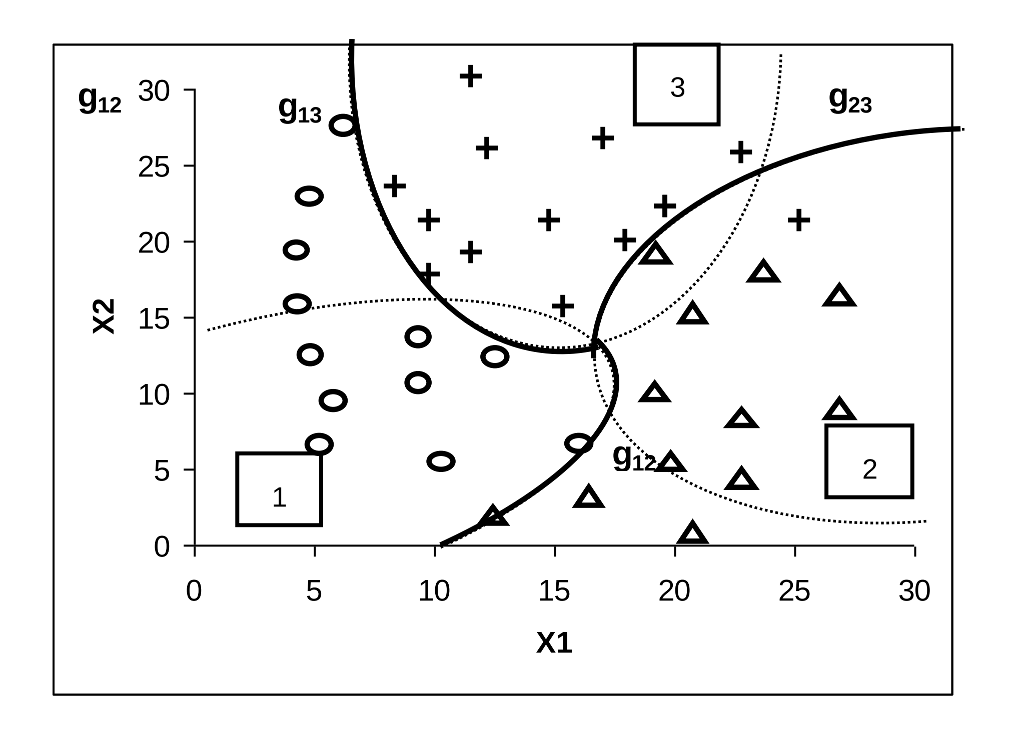
$$g_i(\vec{X}_j^{(i)}) = \max_{k=1,\ldots,M} \{g_k(\vec{X}_j^{(i)})\}, \ \forall j = 1,\ldots,n_i$$

Fronteras de decisión

$$g_{ij}(\vec{X}): \quad lineales$$



$$g_{ij}(\vec{X}): \quad cuadraticas$$



Clasificación con Regresión Lineal: 1

- Para cada clase se define la función de pertenencia gi:

$$g_i(\vec{X}) = \begin{cases} 1; & \vec{X} \in C_i \\ 0; & \vec{X} \notin C_i \end{cases}$$

- Se construye una función lineal que "aproxime" gi:

$$\vec{y}_i = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \mathrm{H}_i = \begin{bmatrix} 1 & \left(\vec{X}_1^{(i)}\right)^t \\ \vdots & \vdots \\ 1 & \left(\vec{X}_{n_i}^{(i)}\right)^t \\ 1 & \left(\vec{X}_1^{(1)}\right)^t \\ \vdots & \vdots \\ 1 & \left(\vec{X}_{n_I}^{(I)}\right)^t \end{bmatrix}; \qquad \vec{A}_i = [H_i^{\,t} H_i]^{-1} H_i^{\,t} \vec{y}_i$$

- Hay que "aprender" M funciones gi

- Otra opción: para cada par de clases, función frontera gij:

$$g_{ij}(\vec{X}) = \begin{cases} +1; & \vec{X} \in C_i \\ -1; & \vec{X} \in C_j \end{cases}$$

- Funciones lineales para todos los pares:

$$\vec{y}_{ij} = \begin{bmatrix} +1 \\ \vdots \\ +1 \\ -1 \\ \vdots \\ -1 \end{bmatrix} \qquad \mathrm{H}_{ij} = \begin{bmatrix} 1 & \left(\vec{X}_1^{(i)}\right)^t \\ \vdots & \vdots \\ 1 & \left(\vec{X}_{n_i}^{(i)}\right)^t \\ 1 & \left(\vec{X}_1^{(j)}\right)^t \\ \vdots & \vdots \\ 1 & \left(\vec{X}_{n_j}^{(j)}\right)^t \end{bmatrix}; \qquad \vec{A}_{ij} = [H_{ij}{}^t H_{ij}]^{-1} H_{ij}{}^t \vec{y}_i$$

- Hay que “aprender” M(M-1)/2 pares gij fronteras posibles

2.4.1. Clasificación bayesiana

aplicación de modelos estadísticos

- **Clasificación con modelo de estructura probabilística conocida**

Clases: {C1, ..., CM}. Se conoce a priori:

  - **Probabilidades de clase: P(Ci)**
  - **Distribuciones de probabilidad condicionadas (parámetros constantes)**

$$F_{\vec{X}}(x_1,...,x_I \mid C_i) = P(X_1 \le x_I,...,X_I \le x_I \mid C_i) = \frac{P(X_1 \le x_1,...,X_I \le x_I, C_i)}{P(C_i)}$$

  - **densidad**

$$f_{\vec{X}}(x_1,...,x_I \mid C_i) = \frac{\partial F_{\vec{X}}(x_1,...,x_I \mid C_i)}{\partial x_1 \ldots \partial x_I}$$

Ej.: distribución normal multivariada

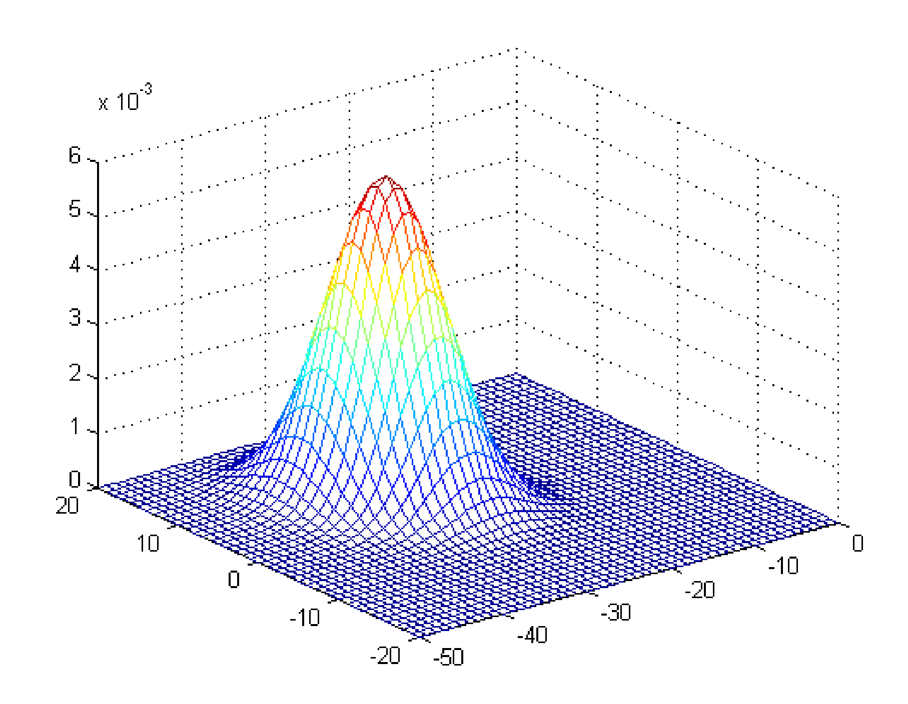
- Parámetros: vector de medias y matriz covarianzas

$$f(\vec{x}) = \frac{1}{(2\pi)^{n/2}\sqrt{|S|}} \exp\left[-\frac{1}{2}(\vec{x}-\vec{\mu})^t S^{-1}(\vec{x}-\vec{\mu})\right]$$

$$\vec{\mu} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix}; \quad S = \begin{bmatrix} \sigma_{x1}^{\ 2} & \sigma_{x_1 x_2} & \cdots & \sigma_{x_1 x_F} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{x_F x_1} & \sigma_{x_n x_2} & \cdots & \sigma_{x_F}^{\ 2} \end{bmatrix}$$

- Ejemplo

$$\vec{\mu} = \begin{bmatrix} -30 \\ 5 \end{bmatrix}; \quad S = \begin{bmatrix} 21 & -6 \\ -6 & 21 \end{bmatrix}$$



Teorema de Bayes aplicado a clasificación

$$P(C_i \mid \vec{X}) = \frac{f(\vec{X} \mid C_i)\, p(C_i)}{f(\vec{X})}$$

- Probabilidad a posteriori: es la probabilidad de que el patrón tenga clase Ci:

$$P(C_i \mid \vec{X})$$

- Probabilidad a priori: P(Ci) es la probabilidad total de cada clase
- Verosimilitud:

$$f(\vec{X} \mid C_i)$$

- : es la distribución de Ci aplicada a
- Densidad total:

$$f(\vec{X}) = f(\vec{X} \mid C_1)P(C_1) + ... + f(\vec{X} \mid C_M)P(C_M)$$

Criterio de clasificación MAP:

$$Clase(\vec{X}) = \underset{i}{máximo}\left\{P(C_i \mid \vec{X})\right\} = \underset{i}{máximo}\left\{f(\vec{X} \mid C_i)\, p(C_i)\right\}$$

  - función discriminante de Ci: proporcional a su prob a posteriori:

$$g_i(\vec{X}) = f(\vec{X} \mid C_i)\, p(C_i)$$

– la clase es la de aquella que maximiza el discriminante

Clasificación bayesiana y distrib. normal

- Distribuciones condicionales gaussianas. Para cada clase Ci hay una función discriminante de parámetros mij, sij, j=1...l

$$g_i(\vec{x}) = \log(P(C_i)\, f(\vec{x} \mid C_i)) = \log \frac{P(C_i)}{(2\pi)^{n/2}\sigma_{1i}\sigma_{2i}...\sigma_{Fi}} - \frac{1}{2}\sum_{i=1}^{F}(x_j - \mu_{ij})^2 / \sigma_{ij}{}^2$$

- Parámetros de distribución condicionada a cada clase
- Regiones de decisión:
  – Funciones cuadráticas (hipérbolas) dadas por diferencias:

$$g_{ij}(\vec{x}) = g_i(\vec{x}) - g_j(\vec{x})$$

  – Si son iguales, y diagonales: regiones lineales (caso particular)

Resumen clasificador bayesiano numérico

- Algoritmo:
  - Estimar parámetros de cada clase Ci (entrenamiento)

$$C_i : \{\vec{X}_1^{(i)},...,\vec{X}_{n_i}^{(i)}\} \longrightarrow \overline{\mu}_i, \quad C_i$$

$$\hat{\vec{\mu}}_i = \frac{1}{n}\sum_{j=1}^{n_i} \vec{x}_j^{(i)}$$

$$C_i = \frac{1}{n_i}\sum_{j=1}^{n_i} (\vec{x}_i - \overline{\mu}_i)^2$$

- Estimar probabilidad de cada clase

$$\hat{P}(C_i) = \frac{n_i}{N}; \qquad n = \sum_{i=1}^{M} n_i$$

- Obtener regiones de decisión: gij(.)

Clasificación Bayesiana con Atributos Nominales

Atributos nominales con valores discretos

- Ai={V1,...,Vni}: atributo con ni valores posibles
- Pasamos de densidades a probabilidades: probabilidad a priori: p(Ai=Vj|Ck)?
- Estimación “contando” el número de casos:

$$p(A_i = V_j \mid C_k) = \frac{n^o \text{ de } ejemplos \text{ de clase } C_k \text{ con } A_i = V_j}{n^o \text{ de } ejemplos \text{ de clase } C_k}$$

- Simplificación: independencia de atributos (“Naive Bayes”): la probabilidad conjunta de varios atributos se pone como producto

$$X_i = (A_1 = V_1, A_2 = V_2, ..., A_I = V_I)$$
$$p(X_i \mid C_k) = p(A_1 = V_1 \mid C_k) * p(A_2 = V_2 \mid C_k * ... * p(A_I = V_I \mid C_k)$$

- Clasificación:

$$p(C_k \mid X_i) = \frac{p(X_i \mid C_k) * p(C_k)}{p(X_i)} =$$
$$\frac{p(A_1 = V_1 \mid C_k) * p(A_2 = V_2 \mid C_k) * ... * p(A_F = V_F \mid C_k) * p(C_k)}{p(X_i)}$$

Ejemplo con atributos nominales

SALARIO	CLIENTE	EDAD	HIJOS	CRÉDITO
Poco	Sí	Joven	Uno	**NO**
Mucho	Si	Joven	Uno	**SI**
Mucho	Si	Joven	Uno	**SI**
Poco	Si	Joven	Uno	**NO**
Mucho	Si	Joven	Dos	**SI**
Poco	Si	Joven	Dos	**NO**
Mucho	Si	Adulto	Dos	**SI**
Mucho	Si	Adulto	Dos	**SI**
Poco	No	Adulto	Dos	**NO**
Mucho	Si	Adulto	Dos	**SI**
Medio	No	Adulto	Tres	**NO**
Mucho	Si	Adulto	Dos	**SI**
Medio	Si	Adulto	Dos	**SI**
Medio	No	Adulto	Tres	**NO**
Medio	No	Adulto	Dos	**SI**
Mucho	No	Mayor	Tres	**NO**
Poco	No	Mayor	Tres	**SI**
Poco	No	Mayor	Tres	**SI**
Mucho	No	Mayor	Tres	**NO**
Mucho	No	Mayor	Tres	**SI**

p(SI) = 12/20

p(NO) = 8/20

Salario \ Crédito	No	Sí
Poco	4/8	2/12
Mucho	2/8	8/12
Medio	2/8	2/12

Cliente \ Crédito	No	Sí
Sí	3/8	8/12
No	5/8	4/12

Edad \ Crédito	No	Sí
Joven	3/8	3/12
Adulto	3/8	6/12
Mayor	2/8	3/12

Hijos \ Crédito	No	Sí
Uno	2/8	2/12
Dos	2/8	7/12
Tres	4/8	3/12

- Ej.: **(salario=poco, cliente=si, edad=adulto, hijos=tres)**

$$p(SI \mid X_i) =$$
$$p(s = poco \mid SI) * p(c = si \mid SI) * p(e = adulto \mid SI) * p(h = tres \mid SI) * p(SI) / p(X_i) =$$
$$2/12 * 8/12 * 6/12 * 3/12 * 12/20 / p(Xi) = 0.0083 / p(Xi)$$

$$p(NO \mid X_i) =$$
$$p(s = poco \mid NO) * p(c = si \mid NO) * p(e = adulto \mid NO) * p(h = tres \mid NO) * p(NO) / p(X_i) =$$
$$4/8 * 3/8 * 3/8 * 4/8 * 8/20 / p(Xi) = 0.0141 / p(Xi)$$

Atributos sin valores

- Si el ejemplo a clasificar no tiene un atributo, simplemente se omite.
  - Ej.: (**salario=poco, cliente=si, edad=?, hijos=3**)

$$p(SI \mid X_i) =$$
$$p(s = poco \mid SI) * p(c = si \mid SI) * p(h = tres \mid SI) * p(SI) / p(X_i) =$$
$$2/12 * 8/12 * 3/12 * 12/20 / p(Xi) = 0.0167 / p(Xi)$$
$$p(NO \mid X_i) =$$
$$p(s = poco \mid NO) * p(c = si \mid NO) * p(h = tres \mid NO) * p(NO) / p(X_i) =$$
$$4/8 * 3/8 * 4/8 * 8/20 / p(Xi) = 0.0375 / p(Xi)$$

- Si hay faltas en la muestra de entrenamiento, no cuentan en la estimación de probabilidades de ese atributo

Faltas en atributo EDAD

SALARIO	CLIENTE	EDAD	HIJOS	CRÉDITO
Poco	Sí	Joven	Uno	NO
Mucho	Si	Joven	Uno	SI
Mucho	Si	Joven	Uno	SI
Poco	Si	?	Uno	NO
Mucho	Si	?	Dos	SI
Poco	Si	?	Dos	NO
Mucho	Si	?	Dos	SI
Mucho	Si	Adulto	Dos	SI
Poco	No	Adulto	Dos	NO
Mucho	Si	Adulto	Dos	SI
Medio	No	Adulto	Tres	NO
Mucho	Si	Adulto	Dos	SI
Medio	Si	Adulto	Dos	SI
Medio	No	Adulto	Tres	NO
Medio	No	Adulto	Dos	SI
Mucho	No	Mayor	Tres	NO
Poco	No	Mayor	Tres	SI
Poco	No	Mayor	Tres	SI
Mucho	No	Mayor	Tres	NO
Mucho	No	Mayor	Tres	SI

p(SI) = 12/20

p(NO) = 8/20



Salario \ Crédito	No	Sí
Poco	4/8	2/12
Mucho	2/8	8/12
Medio	2/8	2/12

Cliente \ Crédito	No	Sí
Sí	3/8	8/12
No	5/8	4/12

Edad \ Crédito	No	Sí
Joven	**1/6**	**2/10**
Adulto	**3/6**	**5/10**
Mayor	**2/6**	**3/10**

Hijos \ Crédito	No	Sí
Uno	2/8	2/12
Dos	2/8	7/12
Tres	4/8	3/12

Atributos no representados. Ley m

- Problema: con muestra poco representativa, puede ocurrir que en alguna clase, un valor de atributo no aparezca: p(Ai=Vj|Ck)=0

   - Cualquier ejemplo X con Ai=Vj generará P(Ck|X)=0, independientemente de los otros atributos!

- Se suele modificar la estimación de las probabilidades a priori con un factor que elimina los ceros.

   - Ej.: P(Edad|Crédito=NO)=

$$\left\{ Joven : \frac{3}{8}, \quad Adulto : \frac{3}{8}, \quad Mayor : \frac{2}{8} \right\}$$

   - Ley m:

$$\left\{ Joven : \frac{3+\mu/3}{8+\mu}, \quad Adulto : \frac{3+\mu/3}{8+\mu}, \quad Mayor : \frac{2+\mu/3}{8+\mu} \right\}$$

   - A veces simplemente se inicializan las cuentas a 1 en vez de 0:

$$\left\{ Joven : \frac{3+1}{8+3}, \quad Adulto : \frac{3+1}{8+3}, \quad Mayor : \frac{2+1}{8+3} \right\}$$

Atributos mixtos

- Independencia de atributos (“Naive Bayes”)

$$\begin{aligned} &\mathrm{p}(\mathrm{X_i} \mid \mathrm{C_k}) = \\ &\mathrm{p}(\mathrm{A_1} = \mathrm{V_1} \mid \mathrm{C_k}) * \mathrm{p}(\mathrm{A_2} = \mathrm{V_2} \mid \mathrm{C_k}) * \ldots * \mathrm{p}(\mathrm{A_F} = \mathrm{V_F} \mid \mathrm{C_k}) \end{aligned}$$

   - Atributos discretos: probabilidades a priori con cada clase Ck

$$p(A_i = V_j \mid C_k) = \frac{n^{\text{o}} \text{ de e}\textit{jemplos}\text{ de clase } \mathrm{C_k} \text{ con } \mathrm{A_i} = \mathrm{V_j}}{n^{\text{o}} \text{ de e}\textit{jemplos}\text{ de clase } \mathrm{C_k}}$$

   - Atributos continuos: densidades de clase Ck: normales de parámetros mk, sk

$$p(A_i = V_j \mid C_k) \rightarrow f_{A_i}(V_j \mid C_k) =$$

$$\frac{1}{\sqrt{2\pi}\sigma_{ik}} \exp\left[-\frac{1}{2} \frac{(V_j - \mu_{ik})^2}{\sigma_{ik}^{\ 2}} \right]$$

Ejemplo con atributos mixtos

SALARIO	CLIENTE	EDAD	HIJOS	CRÉDITO
525	Sí	Joven	1	**NO**
2000	Si	Joven	1	**SI**
2500	Si	Joven	1	**SI**
470	Si	Joven	1	**NO**
3000	Si	Joven	2	**SI**
510	Si	Joven	2	**NO**
2800	Si	Adulto	2	**SI**
2700	Si	Adulto	2	**SI**
550	No	Adulto	2	**NO**
2600	Si	Adulto	2	**SI**
1100	No	Adulto	3	**NO**
2300	Si	Adulto	2	**SI**
1200	Si	Adulto	2	**SI**
900	No	Adulto	3	**NO**
800	No	Adulto	2	**SI**
800	No	Mayor	3	**NO**
1300	No	Mayor	3	**SI**
1100	No	Mayor	3	**SI**
1000	No	Mayor	3	**NO**
4000	No	Mayor	3	**SI**

p(SI) = 12/20

p(NO) = 8/20

Salario \ Crédito	No	Sí
Media	732	2192
Desv Estándar	249	942

Cliente \ Crédito	No	Sí
Sí	3/8	8/12
No	5/8	4/12

Edad \ Crédito	No	Sí
Joven	3/8	3/12
Adulto	3/8	6/12
Mayor	2/8	3/12

Hijos \ Crédito	No	Sí
Media	2.25	2.08
Desv Estándar	0.89	0.67

- Ej.: (salario=700, cliente=si, edad=adulto, hijos=3)

$$
\begin{aligned}
&p(SI \mid X_i) = \\
&\quad f_S(s=700 \mid SI) * p(c=si \mid SI) * p(e=adulto \mid SI) * f_H(h=3 \mid SI) * p(SI) / p(X_i) = \\
&\quad \frac{1}{\sqrt{2\pi}942}\exp\left[-\frac{1}{2}\frac{(700-2192)^2}{942^2}\right] * 8/12 * 6/12 * \frac{1}{\sqrt{2\pi}0.67}\exp\left[-\frac{1}{2}\frac{(3-2.08)^2}{0.67^2}\right] * 12/20 * 1/P(X_i) = \\
&\quad = 5.61e-6 / p(Xi)
\end{aligned}
$$

$$
\begin{aligned}
&p(NO \mid X_i) = \\
&\quad f_S(s=700 \mid NO) * p(c=si \mid NO) * p(e=adulto \mid NO) * f_H(h=3 \mid NO) * p(NO) / p(X_i) = \\
&\quad \frac{1}{\sqrt{2\pi}249}\exp\left[-\frac{1}{2}\frac{(700-732)^2}{249^2}\right] * 3/8 * 3/8 * \frac{1}{\sqrt{2\pi}0.89}\exp\left[-\frac{1}{2}\frac{(3-2.25)^2}{0.89^2}\right] * 8/20 * 1/P(X_i) = \\
&\quad = 2.81e-5 / p(Xi)
\end{aligned}
$$

Clasificación con costes

- MAP proporciona clasificación con mínima prob. de Error
  - Coste de decisión : prob. Error total=
- Con frecuencia los costes son asimétricos, y unos errores son más graves que otros. Matriz de costes

Clasificado como

Clase real

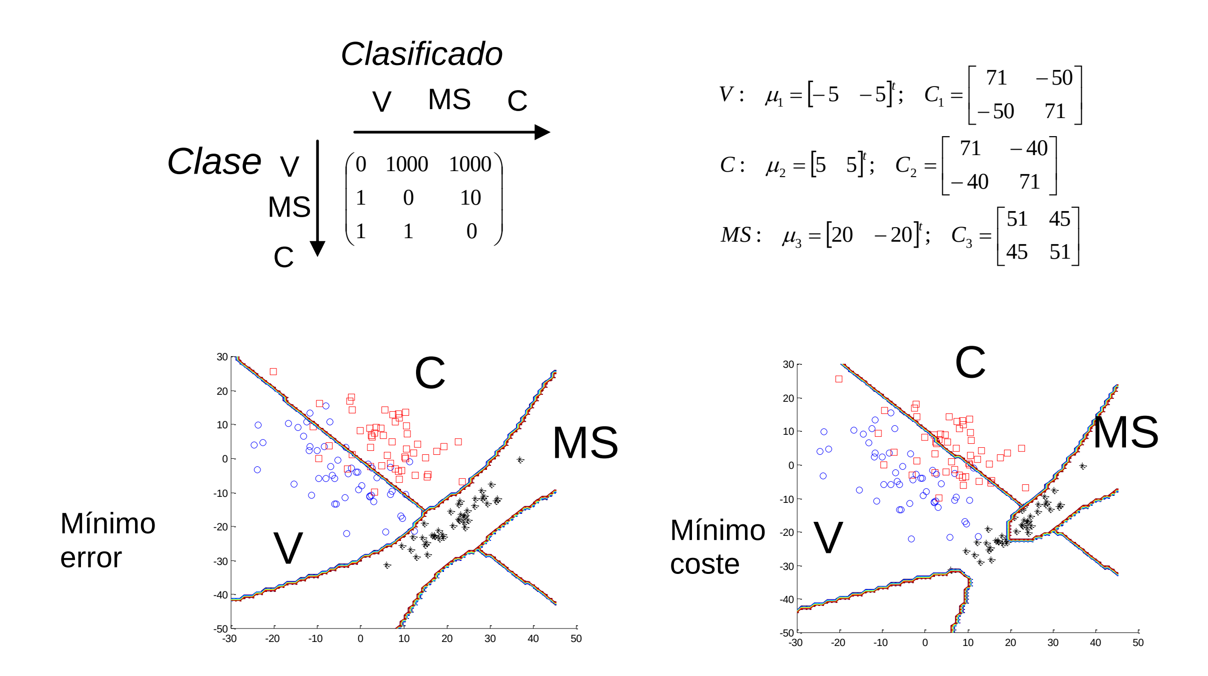
$$
\begin{pmatrix} 0 & c_{12} & c_{13} \\ c_{21} & 0 & c_{23} \\ c_{31} & c_{32} & 0 \end{pmatrix}
$$

- Costes de cada decisión. Criterio de mínimo coste medio: dada una decisión, promedio los costes de cada equivocación y su coste:

$$
\begin{aligned}
\text{coste}(D_1 \mid \vec{X}) &= c_{21} p(C_2 \mid \vec{X}) + c_{31} p(C_3 \mid \vec{X}) \\
\text{coste}(D_2 \mid \vec{X}) &= c_{12} p(C_1 \mid \vec{X}) + c_{32} p(C_3 \mid \vec{X}) \\
\text{coste}(D_3 \mid \vec{X}) &= c_{13} p(C_1 \mid \vec{X}) + c_{23} p(C_2 \mid \vec{X})
\end{aligned}
$$

Ejemplo de clasificación con costes

- Clasificación de setas con dos atributos, (X, Y) y tres categorías: V*enenosa, Mal sabor, comestible*: {*V, MS, C*}



2.4.2. Regresión Lineal

La regresión lineal [DOB90] es la forma más simple de regresión, ya que en ella se modelan los datos usando una línea recta. Se caracteriza, por tanto, por la utilización de dos variables, una aleatoria, *y* (llamada variable respuesta), que es función lineal de otra variable aleatoria, *x* (llamada variable predictora), formándose la ecuación 2.13.

$$y = a + bx \qquad \text{Ec. 2.13}$$

En esta ecuación la variación de *y* se asume que es constante, y *a* y *b* son los coeficientes de regresión que especifican la intersección con el eje de ordenadas, y la pendiente de la recta, respectivamente. Estos coeficientes se calculan utilizando el método de los mínimos cuadrados [PTVF96] que minimizan el error entre los datos reales y la estimación de la línea. Dados *s* ejemplos de datos en forma de puntos (x_1, y_1), (x_2, x_2),..., (x_s, y_s), entonces los coeficientes de la regresión pueden estimarse según el método de los mínimos cuadrados con las ecuaciones 2.14 y 2.15.

$$b = \frac{S_{xy}}{S_x^2} \qquad \text{Ec. 2.14}$$

$$a = y - bx \qquad \text{Ec. 2.15}$$

En la ecuación 2.14, S_{xy} es la covarianza de *x* e *y*, y S_x^2 la varianza de *x*. También es necesario saber cuán buena es la recta de regresión construida. Para ello, se emplea el coeficiente de regresión (ecuación 2.16), que es una medida del ajuste de la muestra.

$$R^2 = \frac{S_{xy}^2}{S_x^2 S_y^2} \qquad \text{Ec. 2.16}$$

El valor de R^2 debe estar entre *0* y *1*. Si se acerca a *0* la recta de regresión no tiene un buen ajuste, mientras que si se acerca a *1* el ajuste es “perfecto”. Los coeficientes *a* y *b* a menudo proporcionan buenas aproximaciones a otras ecuaciones de regresión complicadas.

En el ejemplo siguiente, para una muestra de 35 marcas de cerveza, se estudia la relación entre el grado de alcohol de las cervezas y su contenido calórico. y se representa un pequeño conjunto de datos.

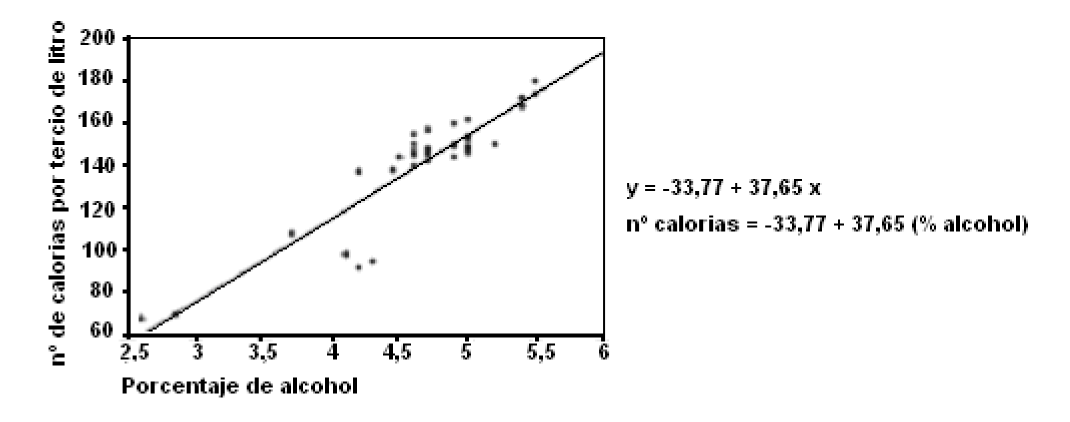


Figura 2.1: Regresión lineal simple.

El eje vertical muestra el número de calorías (por cada tercio de litro) y el horizontal el contenido de alcohol (expresado en porcentaje). La nube de puntos es la representación de los datos de la muestra, y la recta es el resultado de la regresión lineal aplicando el ajuste de los mínimos cuadrados. En los siguientes apartados se mostrarán dos tipos de regresiones que amplían la regresión lineal simple.

- **Regresión Lineal Múltiple**

La regresión Lineal Múltiple [PTVF96] es una extensión de regresión lineal que involucra más de una variable predictora, y permite que la variable respuesta *y* sea planteada como una función lineal de un vector multidimensional. El modelo de regresión múltiple para *n* variables predictoras sería como el que se muestra en la ecuación 2.17.

$$y = b_0 + b_1 x_1 + b_2 x_2 + ... + b_n x_n \quad \text{Ec. 2.17}$$

Para encontrar los coeficientes b_i se plantea el modelo en términos de matrices, como se muestra en la ecuación 2.18.

$$Z = \begin{pmatrix} z_{11} & \cdots & z_{1n} \\ z_{21} & \cdots & z_{1n} \\ \vdots & & \vdots \\ z_{m1} & \cdots & z_{mn} \end{pmatrix};\ Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix};\ B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \quad \text{Ec. 2.18}$$

En la matriz *Z,* las filas representan los *m* ejemplos disponibles para calcular la regresión, y las columnas los *n* atributos que formarán parte de la regresión. De esta forma, z_{ij} será el valor que toma en el ejemplo *i* el atributo *j*. El vector *Y* está formado por los valores de la variable dependiente para cada uno de los ejemplos, y el vector *B* es el que se desea calcular, ya que se corresponde con los parámetros desconocidos necesarios para construir la regresión lineal múltiple. Representando con X^T la matriz traspuesta de *X* y con X^{-1} la inversa de la matriz *X*, se calculará el vector *B* mediante la ecuación 2.19.

$$B = \left(Z^T Z\right)^{-1} Z^T Y \quad \text{Ec. 2.19}$$

Para determinar si la recta de regresión lineal múltiple está bien ajustada, se emplea el mismo concepto que en el caso de la regresión lineal simple: el coeficiente de regresión. En este caso, se utilizará la ecuación 2.20.

$$R^2 = 1 - \frac{(Y - ZB)^T (Y - ZB)^T}{\sum_{i=1}^{m} \left(y_i - \overline{y}\right)^2} \quad \text{Ec. 2.20}$$

Al igual que en el caso de la regresión simple, el valor de R^2 debe estar entre *0* y *1*, siendo *1* el indicador de ajuste perfecto.

Una vez explicado el modo básico por el que se puede obtener una recta de regresión múltiple para un conjunto de ejemplos de entrenamiento, a continuación se muestra, en la figura 2.11, un ejemplo concreto en el que se muestra el proceso.

Paso 1. Ejemplos de Entrenamiento y Matrices

Atributo 1 (X1)	Atributo 2 (X2)	Atributo 3 (X3)	Clase (Y)
1	3	-2	1
2	1	3	-2,3
4	6	5	-10
3	3	2	-1
2	4	-1	0,5
3	2	1	3,1

$$Z=\begin{pmatrix}1&3&-2\\2&1&3\\4&6&5\\3&3&2\\1&4&-1\\3&2&1\end{pmatrix};\ Y=\begin{pmatrix}1\\-2,3\\-10\\-1\\0,5\\3,1\end{pmatrix};\ B=\begin{pmatrix}b1\\b2\\b3\end{pmatrix}$$

Paso 2. Obtención de los Coeficientes y Recta de Regresión

$$B=\left(Z^TZ\right)^{-1}Z^TY=\left(\begin{pmatrix}1&2&4&3&1&3\\3&1&6&3&4&2\\-2&3&5&2&-1&1\end{pmatrix}\begin{pmatrix}1&3&-2\\2&1&3\\4&6&5\\3&3&2\\1&4&-1\\3&2&1\end{pmatrix}\right)^{-1}\begin{pmatrix}1&2&4&3&1&3\\3&1&6&3&4&2\\-2&3&5&2&-1&1\end{pmatrix}\begin{pmatrix}1\\-2,3\\-10\\-1\\0,5\\3,1\end{pmatrix}=$$

$$y=2,608x1-1,494x2-2,169x3$$

$$\begin{pmatrix}40&48&32\\48&75&31\\32&31&44\end{pmatrix}^{-1}\begin{pmatrix}-36,8\\-54,1\\-58,31\end{pmatrix}=\begin{pmatrix}0,22033&-0,1055&-0,0859\\-0,1055&0,06933&0,02788\\-0,0859&0,02788&0,06556\end{pmatrix}\begin{pmatrix}-36,8\\-54,1\\-58,31\end{pmatrix}=\begin{pmatrix}2,608\\-1,494\\-2,169\end{pmatrix}$$

Paso 3. Comprobación del ajuste de la recta de regresión

$$R^2=1-\frac{(Y-ZB)^T(Y-ZB)}{\sum_{i=1}^{m}\left(y_i-\bar{y}\right)^2}=1-\frac{5,841}{104,535}=0,944$$

Figura 2.2: Ejemplo de obtención de una Regresión Lineal Múltiple.

Tal y como se muestra en la figura 2.11, en un primer momento se obtienen, a partir de los ejemplos de entrenamiento, las matrices *Z* e *Y*, siendo el objetivo la matriz *B*. En el segundo paso se calcula los valores de dicha matriz, que serán los coeficientes en la regresión. Por último, en un tercer paso se comprueba si la recta generada tiene un buen ajuste o no. En este caso, como se muestra en la misma figura, el ajuste es magnífico, dado que el valor de R^2 es muy cercano a *1*. Por último, en este ejemplo no se ha considerado el término independiente, pero para que se obtuviese bastaría con añadir una nueva columna a la matriz *Z* con todos los valores a *1*.

Selección de Variables

Además del proceso anterior para la generación de la regresión lineal, se suele realizar un procedimiento estadístico que seleccione las mejores variables predictoras, ya que no todas tienen la misma importancia, y reducir su número hará que computacionalmente mejore el tiempo de respuesta del modelo. Los procesos que se siguen para la selección de variables predictoras son básicamente dos: *eliminación hacia atrás* [backward elimination], consistente en obtener la regresión lineal para todos los parámetros e ir eliminando uno a uno los menos importantes; y *selección hacia delante* [fordward selection], que consiste en generar una regresión lineal simple (con el mejor parámetro, esto es, el más correlacionado con la variable a predecir) e ir añadiendo parámetros al modelo. Hay un gran número de estadísticos que permiten seleccionar los parámetros, y a modo de ejemplo se comentará el basado en el *criterio*

de información Akaike [AKA73], que se basa en la teoría de la información y cuya formulación se muestra en la ecuación 2.21.

$$AIC = -2 \times log(L) + 2p \qquad \text{Ec. 2.21}$$

En esta ecuación *L* es la *verosimilitud* [likelihood] y *p* el número de variables predictorias. Aplicado a la regresión, el resultado sería el que se muestra en las ecuaciones 2.22 y 2.23.

$$AIC = m \times log(MSE) + 2p \qquad \text{Ec. 2.22}$$

$$MSE = \frac{\sum_{i=1}^{m} (y_i - \hat{y}_i)^2}{m} \qquad \text{Ec. 2.23}$$

En la ecuación 2.22, *m* es el número de ejemplos disponibles, y *MSE* es el error cuadrático medio [mean squared error] del modelo, tal y como se define en la ecuación 2.23. En esta ecuación y_i es el valor de la clase para el ejemplo *i* e $\hat{y}_i$ el valor que la regresión lineal da al ejemplo *i*. En la práctica algunas herramientas no utilizan exactamente la ecuación 2.22, sino una aproximación de dicha ecuación.

- **Regresión Lineal Ponderada Localmente**

Otro método de predicción numérica es la regresión lineal ponderada localmente [Locally weighted linear regresión]. Con este método se generan modelos locales durante el proceso de predicción dando más peso a aquellos ejemplares de entrenamiento más cercanos al que hay que predecir. Dicho de otro modo, la construcción del clasificador consiste en el almacenamiento de los ejemplos de entrenamiento, mientras que el proceso de validación o de clasificación de un ejemplo de test consiste en la generación de una regresión lineal específica, esto es, una regresión lineal en la que se da más peso a aquellos ejemplos de entrenamiento cercanos al ejemplo a clasificar. De esta forma, este tipo de regresión está íntimamente relacionado con los algoritmos basados en ejemplares. Para utilizar este tipo de regresión es necesario decidir un esquema de ponderación para los ejemplos de entrenamiento, esto es, decidir cuánto peso se le va a dar a cada ejemplo de entrenamiento para la clasificación de un ejemplo de test. Una medida usual es ponderar el ejemplo de entrenamiento con la inversa de la distancia euclídea entre dicho ejemplo y el de test, tal y como se muestra en ecuación 2.24.

$$\omega_i = \frac{1}{1 + d_{ij}} \qquad \text{Ec. 2.24}$$

En esta ecuación ω_i es el peso que se le otorgará al ejemplo de entrenamiento *i* para clasificar al ejemplo *j*, y d_{ij} será la distancia euclídea de *i* con respecto a *j*.

Más crítico que la elección del método para ponderar es el “parámetro de suavizado” que se utilizará para escalar la función de distancia, esto es, la distancia será multiplicada por la inversa de este parámetro. Si este parámetro es muy pequeño sólo los ejemplos muy cercanos recibirán un gran peso, mientras que si es demasiado grande los ejemplos muy lejanos podrían tener peso. Un modo de asignar un valor a este parámetro es dándole el valor de la distancia del *k*-ésimo vecino más cercano al

ejemplo a clasificar. El valor de *k* dependerá del *ruido* de los datos. Cuanto más ruido, más grande deberá ser *k*. Una ventaja de este método de estimación es que es capaz de aproximar funciones no lineales. Además, se puede actualizar el clasificador (modelo incremental), dado que únicamente sería necesario añadirlo al conjunto de entrenamiento. Sin embargo, como el resto de algoritmos basado en ejemplares, es lento.

Capítulo 3. Técnicas de Minería de Datos basadas en Aprendizaje Automático

3.1. Técnicas de Minería de Datos

Como ya se ha comentado, las técnicas de Minería de Datos (una etapa dentro del proceso completo de KDD [FAYY96]) intentan obtener patrones o modelos a partir de los datos recopilados. Decidir si los modelos obtenidos son útiles o no suele requerir una valoración subjetiva por parte del usuario. Las técnicas de Minería de Datos se clasifican en dos grandes categorías: supervisadas o predictivas y no supervisadas o descriptivas [WI98].

- Técnicas
  - No supervisadas
    - Clustering
      - Numérico
      - Conceptual
      - Probabilistico
    - Asociación
      - A Priori
  -
    - Predicción
      - Regresión
      - Árboles de Predicción
      - Estimador de Núcleos



Figura 3.1: *Técnicas de la Minería de Datos*

Una técnica constituye el enfoque conceptual para extraer la información de los datos, y, en general es implementada por varios algoritmos. Cada algoritmo representa, en la práctica, la manera de desarrollar una determinada técnica paso a paso, de forma que es preciso un entendimiento de alto nivel de los algoritmos para saber cual es la técnica más apropiada para cada problema. Asimismo es preciso entender los parámetros y las características de los algoritmos para preparar los datos a analizar.

Las predicciones se utilizan para prever el comportamiento futuro de algún tipo de entidad mientras que una descripción puede ayudar a su comprensión. De hecho, los modelos predictivos pueden ser descriptivos (hasta donde sean comprensibles por personas) y los modelos descriptivos pueden emplearse para realizar predicciones. De esta forma, hay algoritmos o técnicas que pueden servir para distintos propósitos, por lo que la figura anterior únicamente representa para qué propósito son más utilizadas las técnicas. Por ejemplo, las redes de neuronas pueden servir para predicción, clasificación e incluso para aprendizaje no supervisado.

El aprendizaje inductivo no supervisado estudia el aprendizaje sin la ayuda del *maestr*o; es decir, se aborda el aprendizaje sin supervisión, que trata de ordenar los ejemplos en una jerarquía según las regularidades en la distribución de los pares atributo-valor sin la *guía* del atributo especial *clas*e. Éste es el proceder de los sistemas que realizan *clustering* conceptual y de los que se dice también que adquieren nuevos conceptos. Otra posibilidad contemplada para estos sistemas es la de sintetizar conocimiento cualitativo o cuantitativo, objetivo de los sistemas que llevan a cabo tareas de descubrimiento.

En el aprendizaje inductivo supervisado existe un atributo especial, normalmente denominado *clase,* presente en todos los ejemplos que especifica si el ejemplo pertenece o no a un cierto concepto, que será el objetivo del aprendizaje. El atributo clase normalmente toma los valores + y -, que significan la pertenencia o no del ejemplo al concepto que se trata de aprender; es decir, que el ejemplo ejemplifica positivamente al concepto -pertenece al concepto- o bien lo ejemplifica negativamente -que no pertenece al concepto. Mediante una generalización del papel del atributo clase, cualquier atributo puede desempeñar ese papel, convirtiéndose la *clasificación* de los ejemplos según los valores del atributo en cuestión, en el objeto del aprendizaje. Expresado en una forma breve, el objetivo del aprendizaje supervisado es: a partir de un conjunto de ejemplos, denominados de entrenamiento, de un cierto dominio *D* de ellos, construir criterios para determinar el valor del atributo clase en un ejemplo cualquiera del dominio. Esos criterios están basados en los valores de uno o varios de los otros pares (atributo; valor) que intervienen en la definición de los ejemplos. Es sencillo transmitir esa idea al caso en el que el atributo que juega el papel de la clase sea uno cualquiera o con más de dos valores. Dentro de este tipo de aprendizaje se pueden distinguir dos grandes grupos de técnicas: la predicción y la clasificación [WK91]. A continuación se presentan las principales técnicas (supervisadas y no supervisadas) de minería de datos

3.2. Clustering. (“Segmentación”)

También llamada agrupamiento, permite la identificación de tipologías o grupos donde los elementos guardan gran similitud entre sí y muchas diferencias con los de otros grupos. Así se puede segmentar el colectivo de clientes, el conjunto de valores e índices financieros, el espectro de observaciones astronómicas, el conjunto de zonas forestales, el conjunto de empleados y de sucursales u oficinas, etc. La segmentación está teniendo mucho interés desde hace ya tiempo dadas las importantes ventajas que aporta al permitir el tratamiento de grandes colectivos de forma pseudoparticularizada, en el más idóneo punto de equilibrio entre el tratamiento individualizado y aquel totalmente masificado.

Las herramientas de segmentación se basan en técnicas de carácter estadístico, de empleo de algoritmos matemáticos, de generación de reglas y de redes neuronales para el tratamiento de registros. Para otro tipo de elementos a agrupar o segmentar, como texto y documentos, se usan técnicas de reconocimiento de conceptos. Esta técnica suele servir de punto de partida para después hacer un análisis de clasificación sobre los *clusters*.

La principal característica de esta técnica es la utilización de una medida de similaridad que, en general, está basada en los atributos que describen a los objetos, y se define usualmente por proximidad en un espacio multidimensional. Para datos numéricos, suele ser preciso preparar los datos antes de realizar data mining sobre ellos, de manera que en primer lugar se someten a un proceso de estandarización. Una de las técnicas empleadas para conseguir la normalización de los datos es utilizar la medida z (*z-score*) que elimina las unidades de los datos. Esta medida, z, es la que se muestra en la ecuación 2.1, donde μ_f es la media de la variable *f* y σ_f la desviación típica de la misma.

$$z_{if} = \frac{x_{if} - \mu_f}{\sigma_f}$$ Ec. 2.1

Entre las medidas de similaridad destaca la distancia euclídea, ecuación 2.2.

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^{n} \left(x_{il} - x_{jl}\right)^2}$$ Ec. 2.2

Hay varios algoritmos de *clustering*. A continuación se exponen los más conocidos.

3.2.1. Clustering Numérico (k-medias)

Uno de los algoritmos más utilizados para hacer clustering es el *k*-medias (k-means) [MAC67], que se caracteriza por su sencillez. En primer lugar se debe especificar por adelantado cuantos clusters se van a crear, éste es el parámetro k, para lo cual se seleccionan *k* elementos aleatoriamente, que representaran el centro o media de cada cluster. A continuación cada una de las instancias, ejemplos, es asignada al centro del cluster más cercano de acuerdo con la distancia Euclidea que le separa de él. Para cada uno de los clusters así construidos se calcula el centroide de todas sus instancias. Estos centroides son tomados como los nuevos centros de sus respectivos clusters. Finalmente se repite el proceso completo con los nuevos centros de los clusters. La iteración continúa hasta que se repite la asignación de los mismos ejemplos a los mismos clusters, ya que los puntos centrales de los clusters se han estabilizado y permanecerán invariables después de cada iteración. El algoritmo de *k*-medias es el siguiente:

```
1. Elegir k ejemplos que actúan como semillas (k número de
   clusters).

2. Para cada ejemplo, añadir ejemplo a la clase más similar.

3. Calcular el centroide de cada clase, que pasan a ser las nuevas
   semillas

4. Si no se llega a un criterio de convergencia (por ejemplo, dos
   iteraciones no cambian las clasificaciones de los ejemplos), volver
   a 2.
```

Figura 3.2: *Pseudocódigo del algoritmo de k-medias.*

Para obtener los centroides, se calcula la media [mean] o la moda [mode] según se trate de atributos numéricos o simbólicos. A continuación, en la figura 2.3, se muestra un ejemplo de clustering con el algoritmo *k*-medias.

En este caso se parte de un total de nueve ejemplos o instancias, se configura el algoritmo para que obtenga 3 clusters, y se inicializan aleatoriamente los centroides de los clusters a un ejemplo determinado. Una vez inicializados los datos, se comienza el bucle del algoritmo. En cada una de las gráficas inferiores se muestra un paso por el algoritmo. Cada uno de los ejemplos se representa con un tono de color diferente que indica la pertenencia del ejemplo a un cluster determinado, mientras que los centroides siguen mostrándose como círculos de mayor tamaño y sin relleno. Por ultimo el proceso de clustering finaliza en el paso 3, ya que en la siguiente pasada del algoritmo (realmente haría cuatro pasadas, si se configurara así) ningún ejemplo cambiaría de cluster.

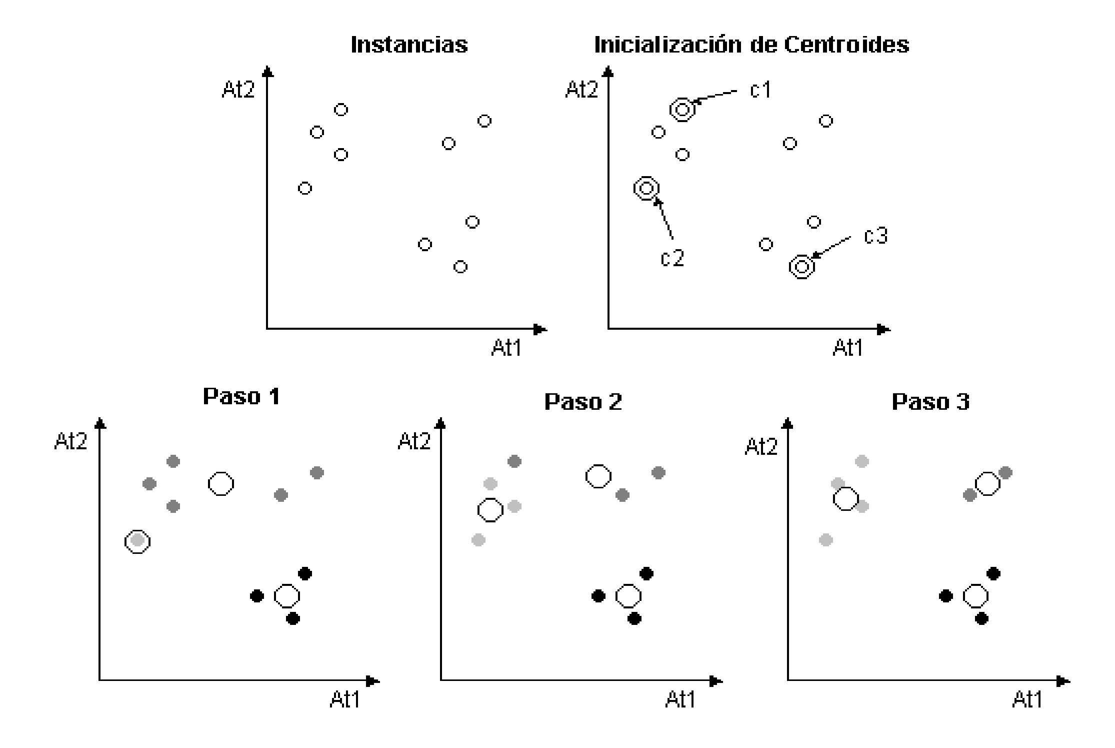


Figura 3.3: *Ejemplo de clustering con k-medias.*

3.2.2. Clustering Conceptual (COBWEB)

El algoritmo de *k*-medias se encuentra con un problema cuando los atributos no son numéricos, ya que en ese caso la distancia entre ejemplares no está tan clara. Para resolver este problema Michalski [MS83] presenta la noción de clustering conceptual, que utiliza para justificar la necesidad de un clustering cualitativo frente al clustering cuantitativo, basado en la vecindad entre los elementos de la población. En este tipo de clustering una partición de los datos es buena si cada clase tiene una

buena interpretación conceptual (modelo cognitivo de jerarquías). Una de las principales motivaciones de la categorización de un conjunto de ejemplos, que básicamente supone la formación de conceptos, es la predicción de características de las categorías que heredarán sus subcategorías. Esta conjetura es la base de COBWEB [FIS87]. A semejanza de los humanos, COBWEB forma los conceptos por agrupación de ejemplos con atributos similares. Representa los clusters como una distribución de probabilidad sobre el espacio de los valores de los atributos, generando un árbol de clasificación jerárquica en el que los nodos intermedios definen subconceptos. El objetivo de COBWEB es hallar un conjunto de clases o clusters (subconjuntos de ejemplos) que maximice la utilidad de la categoría (partición del conjunto de ejemplos cuyos miembros son clases). La descripción probabilística se basa en dos conceptos:

- Predicibilidad: Probabilidad condicional de que un suceso tenga un cierto atributo dada la clase, $P(A_i=V_{ij}|C_k)$. El mayor de estos valores corresponde al valor del atributo más predecible y es el de los miembros de la clase (alta similaridad entre los elementos de la clase).

- Previsibilidad: Probabilidad condicional de que un ejemplo sea una instancia de una cierta clase, dado el valor de un atributo particular, $P(C_k|A_i=V_{ij})$. Un valor alto indica que pocos ejemplos de las otras clases comparten este valor del atributo, y el valor del atributo de mayor probabilidad es el de los miembros de la clase (baja similaridad interclase).

Estas dos medidas, combinadas mediante el teorema de Bayes, proporcionan una función que evalúa la utilidad de una categoría (CU), que se muestra en la ecuación 2.3.

$$CU = \frac{\sum_{k=1}^{n} P(C_k)\left[\sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2\right]}{n} \qquad \text{Ec. 2.3}$$

En esta ecuación n es el número de clases y las sumas se extienden a todos los atributos A_i y sus valores V_{ij} en cada una de las n clases C_k. La división por n sirve para incentivar tener clusters con más de un elemento. La utilidad de la categoría mide el valor esperado de valores de atributos que pueden ser adivinados a partir de la partición sobre los valores que se pueden adivinar sin esa partición. Si la partición no ayuda en esto, entonces no es una buena partición. El árbol resultante de este algoritmo cabe denominarse organización probabilística o jerárquica de conceptos. En la figura 2.4 se muestra un ejemplo de árbol que se podría generar mediante COBWEB. En la construcción del árbol, incrementalmente se incorpora cada ejemplo al mismo, donde cada nodo es un concepto probabilístico que representa una clase de objetos. COBWEB desciende por el árbol buscando el mejor lugar o nodo para cada ejemplo. Esto se basa en medir en cuál se tiene la mayor ganancia de utilidad de categoría.

Datos de Entrada

Nombre	Recubierto de	Cavidades corazón	temperatura del cuerpo	Fertilización
Mamifero	Pelo	Cuatro	Regulada	Interna
Pez	Escamas	Dos	Sin regulación	Externa
Anfibio	Piel húmeda	Tres	Sin regulación	Externa
Ave	Plumas	Cuatro	Regulada	Interna
Reptil	Piel dura	Cuatro Imperfectas	Sin regulación	Interna

Árbol Generado

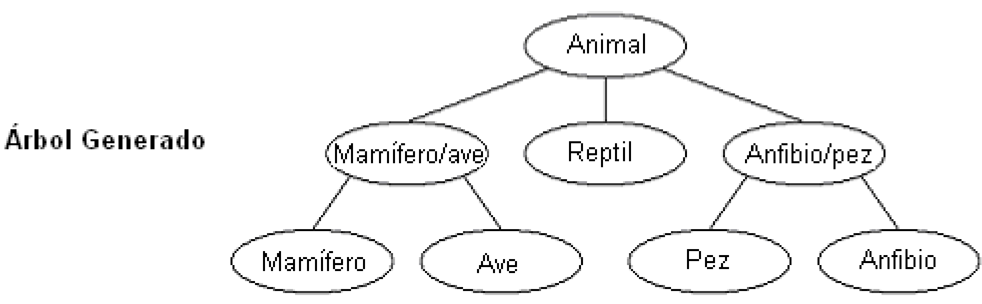


Figura 3.4: Ejemplo de árbol generado por COBWEB.

Sin embargo, no se puede garantizar que se genere este árbol, dado que el algoritmo es sensible al orden en que se introduzcan los ejemplos. En cuanto a las etiquetas de los nodos, éstas fueron puestas a posteriori, coherentes con los valores de los atributos que determinan el nodo. Cuando COBWEB incorpora un nuevo ejemplo en el nodo de clasificación, desciende a lo largo del camino apropiado, actualizando las cuentas de cada nodo, y llevando a cabo por medio de los diferentes operadores, una de las siguientes acciones:

- Incorporación: Añadir un nuevo ejemplo a un nodo ya existente.
- Creación de una nueva disyunción: Crear una nueva clase.
- Unión: Combinar dos clases en una sola.
- División: Dividir una clase existente en varias clases.

La búsqueda, que se realiza en el espacio de conceptos, es por medio de un heurístico basado en el método de escalada gracias a los operadores de unión y división. En la figura 2.5 se muestra el resultado de aplicar cada una de estas operaciones.

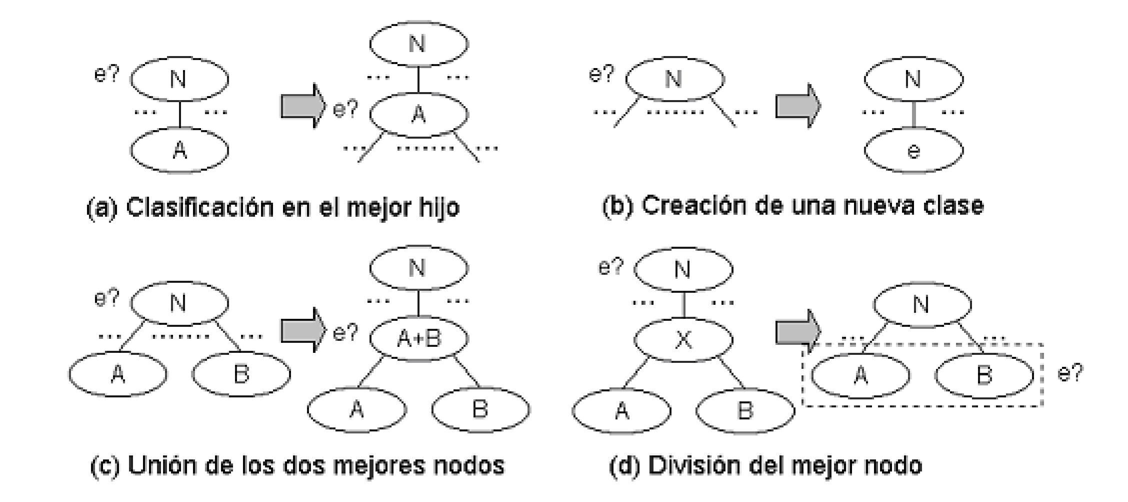


Figura 3.5: *Operaciones de COBWEB.*

1. Nuevo Ejemplo: Lee un ejemplo *e*. Si no hay más ejemplos, terminar.

2. Actualiza raíz. Actualiza el *cálculo* de la raíz.

3. Si la raíz es hoja, entonces: Expandir en dos nodos hijos y acomodar en cada uno de ellos un ejemplo; volver a 1.

4. Avanzar hasta el siguiente nivel: Aplicar la función de evaluación a varias opciones para determinar, mediante la fórmula de utilidad de una categoría, el *mejor* (máxima *CU*) lugar donde incorporar el ejemplo en el nivel siguiente de la jerarquía. En las opciones que se evaluarán se considerará únicamente el nodo actual y sus hijos y se elegirá la *mejor* opción de las siguientes:

   a. Añadir *e* a un nodo que existe (al mejor hijo) y, si esta opción resulta ganadora, comenzar de nuevo el proceso de avance hacia el siguiente nivel en ese nodo hijo.

   b. Crear un nuevo nodo conteniendo únicamente a *e* y, si esta opción resulta ganadora, volver a 1.

   c. Juntar los dos *mejores* nodos hijos con *e* incorporado al nuevo nodo combinado y, si esta opción resulta ganadora, comenzar el nuevo proceso de avanzar hacia el siguiente nivel en ese nuevo nodo.

   d. Dividir el *mejor* nodo, reemplazando este nodo con sus hijos y, si esta opción resulta ganadora, aplicar la función de evaluación para incorporar *e* en los nodos originados por la división.

Figura 3.6: Algoritmo de COBWEB.

El algoritmo se puede extender a valores numéricos usando distribuciones gaussianas, ecuación 2.4. De esta forma, el sumatorio de probabilidades es ahora como se muestra en la ecuación 2.5.

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad \text{Ec. 2.4}$$

$$\sum_j P(A_i=V_{ij})^2 \leftrightarrow \int_{-\infty}^{+\infty} f(x_i)^2\,dx_i = \frac{1}{2\sqrt{\pi}\sigma_i} \qquad \text{Ec. 2.5}$$

Por lo que la ecuación de la utilidad de la categoría quedaría como se muestra en la ecuación 2.6.

$$CU=\frac{1}{k}\sum_{k=1}^{n}P(C_k)\frac{1}{2\sqrt{\pi}}\sum_i\left(\frac{1}{\sigma_{ik}}-\frac{1}{\sigma_i}\right) \qquad \text{Ec. 2.6}$$

3.2.3. Clustering Probabilístico (EM)

Los algoritmos de clustering estudiados hasta el momento presentan ciertos defectos entre los que destacan la dependencia que tiene el resultado del orden de los ejemplos y la tendencia de estos algoritmos al sobreajuste [overfitting]. Una aproximación estadística al problema del clustering resuelve estos problemas. Desde este punto de vista, lo que se busca es el grupo de clusters más probables dados los datos. Ahora los ejemplos tienen ciertas probabilidades de pertenecer a un cluster. La base de este tipo de clustering se encuentra en un modelo estadístico llamado mezcla de distribuciones [finite mixtures]. Cada distribución representa la probabilidad de que un objeto tenga un conjunto particular de pares atributo-valor, si se *supiera* que es miembro de ese cluster. Se tienen *k* distribuciones de probabilidad que representan los *k* clusters. La mezcla más sencilla se tiene cuando los atributos son numéricos con distribuciones gaussianas. Cada distribución (normal) se caracteriza por dos parámetros: la media (μ) y la varianza (σ^2). Además, cada distribución tendrá cierta probabilidad de aparición *p*, que vendrá determinada por la proporción de ejemplos que pertenecen a dicho cluster respecto del número total de ejemplos. En ese caso, si hay *k* clusters, habrá que calcular un total de *3k-1* parámetros: las *k* medias, *k* varianzas y *k-1* probabilidades de la distribución dado que la suma de probabilidades debe ser 1, con lo que conocidas *k-1* se puede determinar la *k-ésima*.

Si se conociera el cluster al que pertenece, en un principio, cada uno de los ejemplos de entrenamiento sería muy sencillo obtener los *3k-1* parámetros necesarios para definir totalmente las distribuciones de dichos clusters, ya que simplemente se aplicarían las ecuaciones de la media y de la varianza para cada uno de los clusters. Además, para calcular la probabilidad de cada una de las distribuciones únicamente se dividiría el número de ejemplos de entrenamiento que pertenecen al cluster en cuestión entre el número total de ejemplos de entrenamiento. Una vez obtenidos estos parámetros, si se deseara calcular la probabilidad de pertenencia de un determinado ejemplo de test a cada cluster, simplemente se aplicaría el teorema de Bayes, ecuación 2.54 a cada problema concreto, con lo que quedaría la ecuación 2.7.

$$P(A \mid x) = \frac{P(x \mid A)P(A)}{P(x)} = \frac{f(x;\mu_A,\sigma_A)p_A}{P(x)} \qquad \text{Ec. 2.7}$$

En esta ecuación A es un cluster del sistema, x el ejemplo de test, p_A la probabilidad del cluster A y $f(x;\mu_A,\sigma_A)$ la función de la distribución normal del cluster A, que se expresa con la ecuación 2.4. Sin embargo, el problema es que no se sabe de qué distribución viene cada dato y se desconocen los parámetros de las distribuciones. Por ello se adopta el procedimiento empleado por el algoritmo de clustering *k*-medias, y se itera.

El algoritmo EM (*Expectation Maximization*) empieza *adivinando* los parámetros de las distribuciones (dicho de otro modo, se empieza *adivinando* las probabilidades de que un objeto pertenezca a una clase) y, a continuación, los utiliza para calcular las probabilidades de que cada objeto pertenezca a un cluster y usa esas probabilidades para re-estimar los parámetros de las probabilidades, hasta converger. Este algoritmo recibe su nombre de los dos pasos en los que se basa cada iteración: el cálculo de las probabilidades de los grupos o los valores esperados de los grupos, mediante la ecuación 2.7, denominado *expectation*; y el cálculo de los valores de los parámetros de las distribuciones, denominado *maximization*, en el que se maximiza la verosimilitud de las distribuciones dados los datos.

Para estimar los parámetros de las distribuciones se tiene que considerar que se conocen únicamente las probabilidades de pertenencia a cada cluster, y no los clusters en sí. Estas probabilidades actúan como pesos, con lo que el cálculo de la media y la varianza se realiza con las ecuaciones 2.8 y 2.9 respectivamente.

$$\mu_A = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i} \qquad \text{Ec. 2.8}$$

$$\sigma_A^2 = \frac{\sum_{i=1}^{N} w_i (x_i - \mu)}{\sum_{i=1}^{N} w_i} \qquad \text{Ec. 2.9}$$

Donde N es el número total de ejemplos del conjunto de entrenamiento y w_i es la probabilidad de que el ejemplo i pertenezca al cluster A. La cuestión es determinar cuándo se finaliza el procedimiento, es decir en que momento se dejan de realizar iteraciones. En el algoritmo *k-medias* se finalizaba cuando ningún ejemplo de entrenamiento cambiaba de cluster en una iteración, alcanzándose así un “punto fijo” [fixed point]. En el algoritmo EM es un poco más complicado, dado que el algoritmo tiende a converger pero nunca se llega a ningún punto fijo. Sin embargo, se puede ver cuánto se acerca calculando la *verosimilitud* [likelihood] general de los datos con esos parámetros, multiplicando las probabilidades de los ejemplos, tal y como se muestra en la ecuación 2.10.

$$\prod_{i=1}^{N} \left(\sum_{j}^{clusters} p_j \; P(x_i \mid j) \right) \qquad \text{Ec. 2.10}$$

En esta ecuación j representa cada uno de los clusters del sistema, y p_j la probabilidad de dicho cluster. La *verosimilitud* es una medida de lo “bueno” que es el clustering, y se incrementa con cada iteración del algoritmo EM. Se seguirá iterando hasta que dicha medida se incremente un valor despreciable.

Aunque EM garantiza la convergencia, ésta puede ser a un máximo local, por lo que se recomienda repetir el proceso varias veces, con diferentes parámetros iniciales para las distribuciones. Tras estas repeticiones, se pueden comparar las medidas de *verosimilitud* obtenidas y escoger la mayor de todas ellas. En la figura 2.7 se muestra un ejemplo de clustering probabilístico con el algoritmo EM.

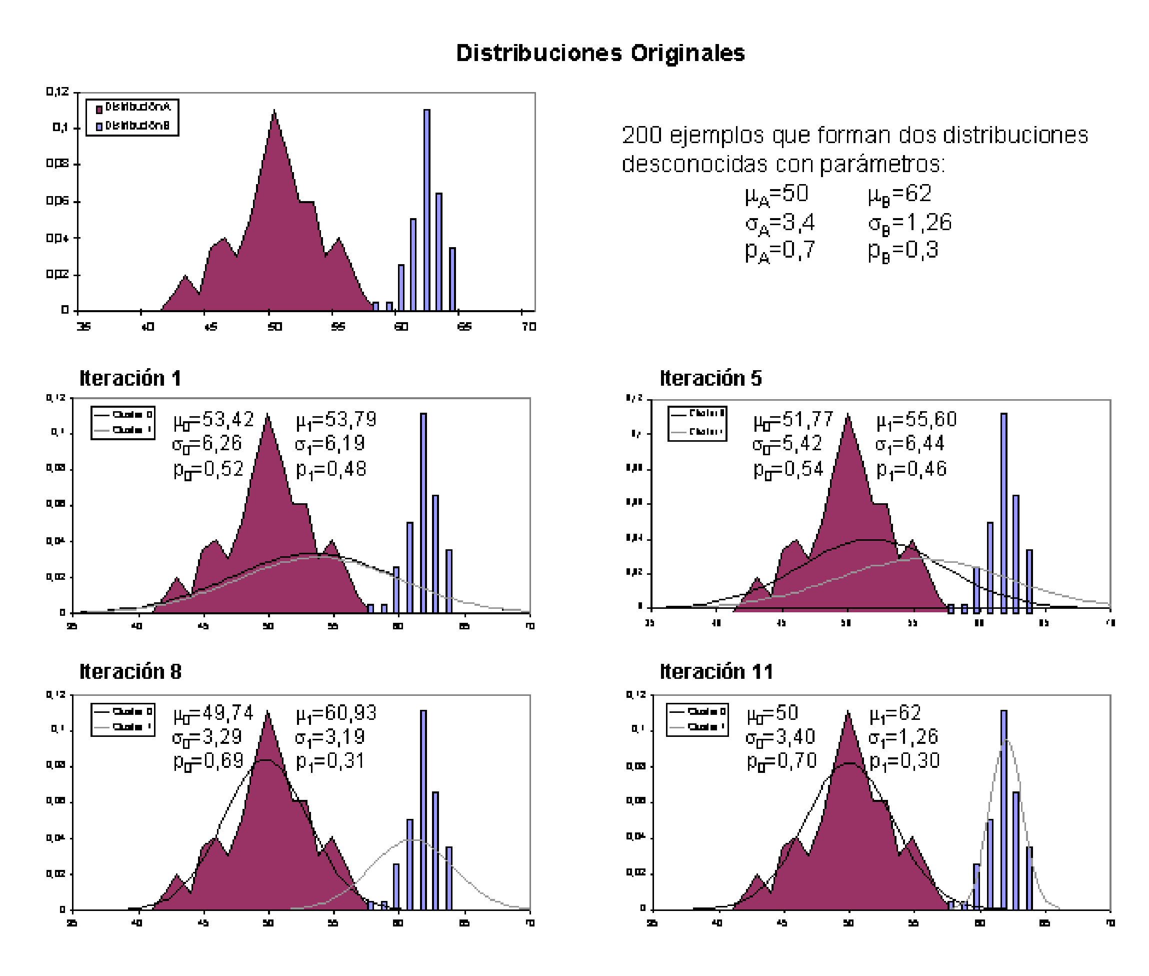


Figura 3.7: Ejemplo de clustering con EM.

En este experimento se introducen un total de doscientos ejemplos que constituyen dos distribuciones desconocidas para el algoritmo. Lo único que conoce el algoritmo es que hay dos clusters, dado que este dato se introduce como parámetro de entrada. En la iteración *0* se inicializan los parámetros de los clusters a *0* (media, desviación típica y probabilidad). En las siguientes iteraciones estos parámetros van tomando forma hasta finalizar en la iteración *11*, iteración en la que finaliza el proceso, por el incremento de la medida de *verosimilitud*, tan sólo del orden de 10^{-4}.

- **Extensiones al algoritmo EM**

El modelo puede extenderse desde un atributo numérico como se ha visto hasta el momento, hasta múltiples atributos, asumiendo independencia entre atributos. Las probabilidades de cada atributo se multiplican entre sí para obtener una probabilidad conjunta para la instancia, tal y como se hace en el algoritmo *naive* Bayesiano. También puede haber atributos correlacionados, en cuyo caso se puede modelar con una distribución normal bivariable, en donde se utiliza una matriz de covarianza. En este caso el número de parámetros crece según el cuadrado del número de atributos que se consideren correlacionados entre sí, ya que se debe construir una matriz de covarianza. Esta escalabilidad en el número de parámetros tiene serias consecuencias de sobreajuste.

En el caso de un atributo nominal con *v* posibles valores, se caracteriza mediante *v* valores numéricos que representan la probabilidad de cada valor. Se necesitarán otros *kv* valores numéricos, que serán las probabilidades condicionadas de cada posible valor del atributo con respecto a cada cluster. En cuanto a los valores desconocidos, se puede optar por varias soluciones: ignorarlo en el productorio de probabilidades; añadir un nuevo valor a los posibles, sólo en el caso de atributos nominales; o tomar la media o la moda del atributo, según se trate de atributos numéricos o nominales. Por último, aunque se puede especificar el número de clusters, también es posible dejar que sea el algoritmo el que determine automáticamente cuál es el número de clusters mediante validación cruzada.

3.3. Reglas de Asociación

Este tipo de técnicas se emplea para establecer las posibles relaciones o correlaciones entre distintas acciones o sucesos aparentemente independientes; pudiendo reconocer como la ocurrencia de un suceso o acción puede inducir o generar la aparición de otros [AIS93b]. Son utilizadas cuando el objetivo es realizar *análisis exploratorios*, buscando relaciones dentro del conjunto de datos. Las asociaciones identificadas pueden usarse para predecir comportamientos, y permiten descubrir correlaciones y co-ocurrencias de eventos [AS94, AS94a, AS94b]. Debido a sus características, estas técnicas tienen una gran aplicación práctica en muchos campos como, por ejemplo, el comercial ya que son especialmente interesantes a la hora de comprender los hábitos de compra de los clientes y constituyen un pilar básico en la concepción de las ofertas y ventas cruzada, así como del "merchandising" [RMS98]. En otros entornos como el sanitario, estas herramientas se emplean para identificar factores de riesgo en la aparición o complicación de enfermedades. Para su utilización es necesario disponer de información de cada uno de los sucesos llevados a cabo por un mismo individuo o cliente en un determinado período temporal. Por lo general esta forma de extracción de conocimiento se fundamenta en técnicas estadísticas [CHY96], como los análisis de correlación y de variación [BMS97]. Uno de los algoritmos mas utilizado es el algoritmo *A priori,* que se presenta a continuación.

Algoritmo A Priori

La generación de reglas de asociación se logra basándose en un procedimiento de *covering*. Las reglas de asociación son parecidas, en su forma, a las reglas de clasificación, si bien en su lado derecho puede aparecer cualquier par o

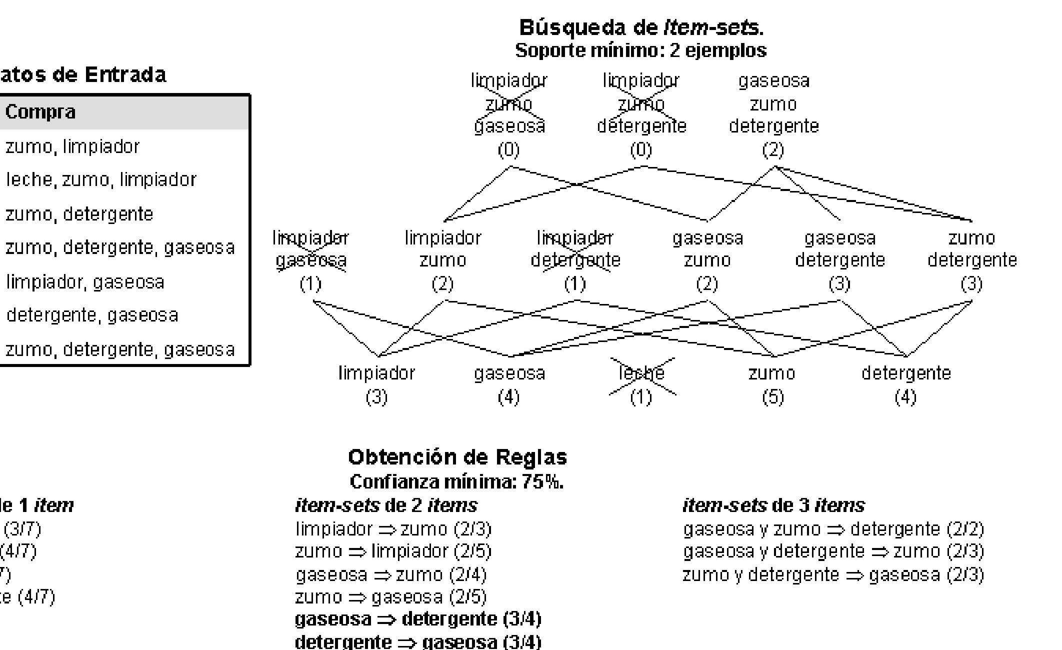
pares *atributo-valor*. De manera que para encontrar ese tipo de reglas es preciso considerar cada posible combinación de pares *atributo-valor* del lado derecho. Para evaluar las reglas se emplean la medida del soporte [support], ecuación 2.11, que indica el número de casos, ejemplos, que cubre la regla y la confianza [confidence], ecuación 2.12, que indica el número de casos que predice la regla correctamente, y que viene expresado como el cociente entre el número de casos en que se cumple la regla y el número de casos en que se aplica, ya que se cumplen las premisas.

$$soporte(A \Rightarrow B) = P(A \cap B) \quad \text{Ec. 2.11}$$

$$confianza(A \Rightarrow B) = P(B \mid A) = \frac{P(A \cap B)}{P(A)} \quad \text{Ec. 2.12}$$

Las reglas que interesan son únicamente aquellas que tienen su valor de soporte muy alto, por lo que se buscan, independientemente de en qué lado aparezcan, pares *atributo-valor* que cubran una gran cantidad de ejemplos. A cada par *atributo-valor* se le denomina *item*, mientras que a un conjunto de *items* se les denomina *item-sets*. Por supuesto, para la formación de *item-sets* no se pueden unir *items* referidos al mismo atributo pero con distinto valor, dado que eso nunca se podría producir en un ejemplo. Se buscan *item-sets* con un máximo soporte, para lo que se comienza con *item-sets* con un único *item*. Se eliminan los *item-sets* cuyo valor de soporte sea inferior al mínimo establecido, y se combinan el resto formando *item-sets* con dos *items*. A su vez se eliminan aquellos nuevos *item-sets* que no cumplan con la condición del soporte, y al resto se le añadirá un nuevo *item*, formando *item-sets* con tres *items*. El proceso continuará hasta que ya no se puedan formar *item-sets* con un *item* más. Además, para generar los *item-sets* de un determinado nivel, sólo es necesario emplear los *item-sets* del nivel inferior (con *n-1* coincidencias, siendo *n* el número de *items* del nivel). Una vez se han obtenido todos los *item-sets*, se pasará a la generación de reglas. Se tomará cada *item-set* y se formarán reglas que cumplan con la condición de *confianza*. Debe tenerse en cuenta que un *item-set* puede dar lugar a más de una regla de asociación, al igual que un *item-set* también puede no dar lugar a ninguna regla.

Un ejemplo típico de reglas de asociación es el análisis de la cesta de la compra [market-basket analysis]. Básicamente consiste en encontrar asociaciones entre los productos que habitualmente compran los clientes, para utilizarlas en el desarrollo de las estrategias mercadotécnicas. En la figura 2.8 se muestra un ejemplo sencillo de obtención de reglas de asociación aplicado a este campo.



Cliente	Compra
1	zumo, limpiador
2	leche, zumo, limpiador
3	zumo, detergente
4	zumo, detergente, gaseosa
5	limpiador, gaseosa
6	detergente, gaseosa
7	zumo, detergente, gaseosa

Figura 3.8: Ejemplo de obtención de reglas de asociación A Priori.

En esta imagen se muestra cómo se forman los *item-sets* a partir de los *item-sets* del nivel inferior, y cómo posteriormente se obtienen las reglas de asociación a partir de los *item-sets* seleccionados. Las reglas en negrita son las que se obtendrían, dado que cumplen con la confianza mínima requerida. El proceso de obtención de las reglas de asociación que se comentó anteriormente se basa en el algoritmo que se muestran en la figura 2.9 (A priori, Agrawal et al. 94).

1. Genera todos los *items-sets* con un elemento. Usa éstos para generar los de dos elementos y así sucesivamente. Se toman todos los posibles pares que cumplen con las medidas mínimas del soporte. Esto permite ir eliminando posibles combinaciones ya que no todas se tienen que considerar.

2. Genera las reglas revisando que cumplan con el criterio mínimo de confianza.

Figura 3.9: Algoritmo de obtención de reglas de asociación A Priori.

Una observación interesante es que si una conjunción de consecuentes de una regla cumple con los niveles mínimos de soporte y confianza, sus subconjuntos (consecuentes) también los cumplen. Por el contrario, si algún *item* no los cumple, no tiene caso considerar sus superconjuntos. Esto da una forma de ir construyendo reglas, con un solo consecuente, y a partir de ellas construir de dos consecuentes y así sucesivamente.

3.4. La predicción

Es el proceso que intenta determinar los valores de una o varias variables, a partir de un conjunto de datos. La predicción de valores continuos puede planificarse por las técnicas estadísticas de regresión [JAM85, DEV95, AGR96]. Por ejemplo, para predecir el sueldo de un graduado de la universidad con 10 años de experiencia de trabajo, o las ventas potenciales de un nuevo producto dado su precio. Se pueden resolver muchos problemas por medio de la regresión lineal, y puede conseguirse todavía más aplicando las transformaciones a las variables para que un problema no lineal pueda convertirse a uno lineal. A continuación se presenta una introducción intuitiva de las ideas de regresión lineal, múltiple, y no lineal, así como la generalización a los modelos lineales.

Más adelante, dentro de la clasificación, se estudiarán varias técnicas de data mining que pueden servir para predicción numérica. De entre todas ellas las más importantes se presentaran en la clasificación bayesiana, la basada en ejemplares y las redes de neuronas. A continuación se mostrarán un conjunto de técnicas que específicamente sirven para la predicción.

3.4.1. Regresión no lineal.

En muchas ocasiones los datos no muestran una dependencia lineal [FRI91]. Esto es lo que sucede si, por ejemplo, la variable respuesta depende de las variables independientes según una función polinómica, dando lugar a una regresión polinómica que puede planearse agregando las condiciones polinómicas al modelo lineal básico. De está forma y aplicando ciertas transformaciones a las variables, se puede convertir el modelo no lineal en uno lineal que puede resolverse entonces por el método de mínimos cuadrados. Por ejemplo considérese una relación polinómica cúbica dada por:

$$y = a + b_1 x + b_2 x^2 + b_3 x^3. \qquad \text{Ec. 2.25}$$

Para convertir esta ecuación a la forma lineal, se definen las nuevas variables:

$$x_1 = x \qquad x_2 = x^2 \qquad x_3 = x^3 \qquad \text{Ec. 2.26}$$

Con lo que la ecuación anterior puede convertirse entonces a la forma lineal aplicando los cambios de variables, y resultando la ecuación 2.27, que es resoluble por el método de mínimos cuadrados

$$y = a + b_1 x_1 + b_2 x_2 + b_3 x_3 \qquad \text{Ec. 2.27}$$

. No obstante, algunos modelos son especialmente no lineales como, por ejemplo, la suma de términos exponenciales y no pueden convertirse a un modelo lineal. Para estos casos, puede ser posible obtener las estimaciones del mínimo cuadrado a través de cálculos extensos en formulas más complejas.

Los modelos lineales generalizados representan el fundamento teórico en que la regresión lineal puede aplicarse para modelar las categorías de las variables dependientes. En los modelos lineales generalizados, la variación de la variable *y* es una función del valor medio de *y*, distinto a la regresión lineal dónde la variación de *y* es constante. Los tipos comunes de modelos lineales generalizados incluyen regresión logística y regresión del Poisson. La regresión logística modela la probabilidad de algún evento que ocurre como una función lineal de un conjunto de variables independientes. Frecuentemente los datos exhiben una distribución de Poisson y se modelan normalmente usando la regresión del Poisson.

Los modelos lineales logarítmicos [PEA88] aproximan las distribuciones de probabilidad multidimensionales discretas, y pueden usarse para estimar el valor de probabilidad asociado con los datos de las células cúbicas. Por ejemplo, suponiendo que se tienen los datos para los atributos ciudad, artículo, año, y ventas. En el método logarítmico lineal, todos los atributos deben ser categorías; por lo que los atributos estimados continuos (como las ventas) deben ser previamente discretizados.

3.4.2. Árboles de Predicción

Los árboles de predicción numérica son similares a los árboles de decisión, que se estudiarán más adelante, excepto en que la clase a predecir es continua. En este caso, cada nodo hoja almacena un valor de clase consistente en la media de las instancias que se clasifican con esa hoja, en cuyo caso estamos hablando de un *árbol de regresión*, o bien un modelo lineal que predice el valor de la clase, y se habla de *árbol de modelos*. En el caso del algoritmo *M5* [WF00], se trata de obtener un árbol de modelos, si bien se puede utilizar para obtener un árbol de regresión, por ser éste un caso específico de árbol de modelos.

Mientras que en el caso de los árboles de decisión se emplea la entropía de clases para definir el atributo con el que dividir, en el caso de la predicción numérica se emplea la *varianza del error* en cada hoja. Una vez construido el árbol que clasifica las instancias se realiza la poda del mismo, tras lo cual, se obtiene para cada nodo hoja una constante en el caso de los árboles de regresión o un plano de regresión en el caso de árboles de modelos. En éste último caso, los atributos que formarán parte de la regresión serán aquellos que participaban en el subárbol que ha sido podado.

Al construir un árbol de modelos y definir, para cada hoja, un modelo lineal con los atributos del subárbol podado suele ser beneficioso, sobre todo cuando se tiene un pequeño conjunto de entrenamiento, realizar un proceso de *suavizado* [smoothing] que compense las discontinuidades que ocurren entre modelos lineales adyacentes. Este proceso consiste en: cuando se predice el valor de una instancia de test con el modelo lineal del nodo hoja correspondiente, este valor obtenido se filtra hacia atrás hasta el nodo hoja, *suavizando* dicho valor al combinarlo con el modelo lineal de cada nodo interior por el que pasa. Un modelo que se suele utilizar es el que se muestra en la ecuación 2.28.

$$p' = \frac{np + kq}{n + k}$$ Ec. 2.28

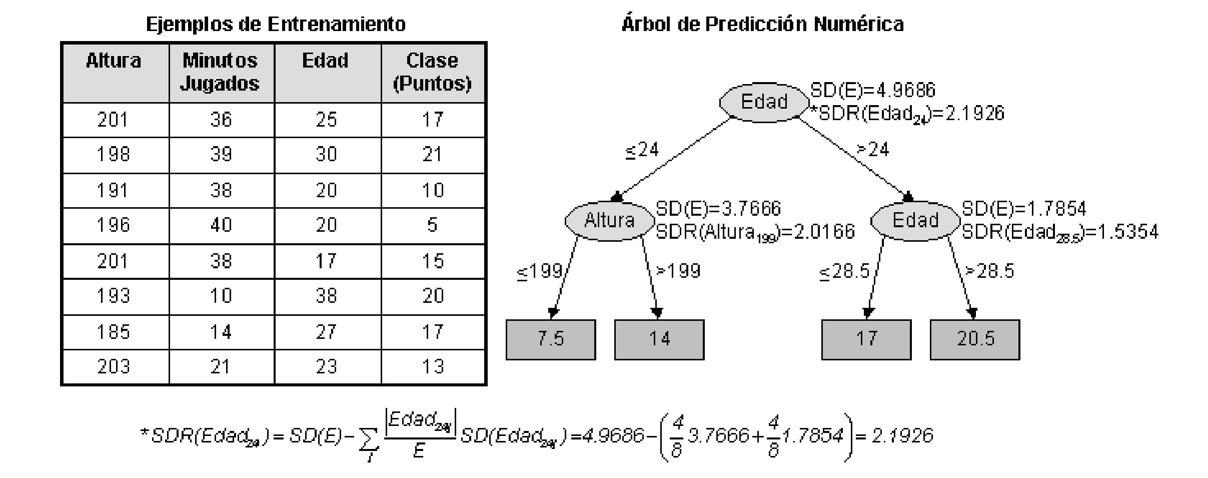
En esta ecuación, *p* es la predicción que llega al nodo (desde abajo), *p'* es la predicción filtrada hacia el nivel superior, *q* el valor obtenido por el modelo lineal de este nodo, *n* es el número de ejemplos que alcanzan el nodo inferior y *k* el factor de suavizado.

Para construir el árbol se emplea como heurística el minimizar la variación interna de los valores de la clase dentro de cada subconjunto. Se trata de seleccionar aquel atributo que maximice la reducción de la desviación estándar de error (SDR, [standard deviation reduction]) con la fórmula que se muestra en la ecuación 2.29.

$$SDR = SD(E) - \sum_i \frac{|E_i|}{|E|} SD(E_i)$$ Ec. 2.29

En esta ecuación *E* es el conjunto de ejemplos en el nodo a dividir, E_j es cada uno de los conjuntos de ejemplos que resultan en la división en el nodo según el atributo considerado, |*E*| es el número de ejemplos del conjunto *E* y *SD(E)* la desviación típica de los valores de la clase en *E*. El proceso de división puede finalizar porque la desviación típica es una pequeña fracción (por ejemplo, el 5%) de la desviación típica del conjunto original de instancias o porque hay pocas instancias (por ejemplo, 2).

En la figura 2.12 se muestra un ejemplo de generación del árbol de predicción con el algoritmo *M5*. Para ello se muestra en primer lugar los ejemplos de entrenamiento, en los que se trata de predecir los puntos que un jugador de baloncesto anotaría en un partido.



Ejemplos de Entrenamiento

Altura	Minutos Jugados	Edad	Clase (Puntos)
201	36	25	17
198	39	30	21
191	38	20	10
196	40	20	5
201	38	17	15
193	10	38	20
185	14	27	17
203	21	23	13

$$*SDR(Edad_{24}) = SD(E) - \sum_i \frac{|Edad_{24i}|}{E} SD(Edad_{24i}) = 4.9686 - \left(\frac{4}{8}3.7666 + \frac{4}{8}1.7854\right) = 2.1926$$

Figura 3.10: Ejemplo de generación del árbol de predicción con M5.

En cada nodo del árbol se muestra la desviación típica de los ejemplos de entrenamiento que inciden en el nodo (*SD(E)*) y la desviación estándar del error para el atributo y el punto de corte que lo maximiza, por lo que es el seleccionado. Para obtener el atributo y el punto de corte se debe calcular la desviación estándar del error para cada posible punto de corte. En este caso, la finalización de la construcción del árbol ocurre porque no se puede seguir subdividiendo, ya que en cada hoja hay dos ejemplos (número mínimo permitido). Por último, tras generar el árbol, en cada hoja se añade la media de los valores de la clase de los ejemplos que se clasifican a través de dicha hoja. Una vez se ha construido el árbol se va definiendo, para cada nodo interior (no para las hojas para emplear el proceso de *suavizado*) un modelo lineal, concretamente una regresión lineal múltiple, tal y como se mostró anteriormente. Únicamente se emplean para realizar esta regresión aquellos atributos que se utilizan en el subárbol del nodo en cuestión.

A continuación se pasa al proceso de poda, en el que se estima, para cada nodo, el error esperado en el conjunto de test. Para ello, lo primero que se hace es calcular la desviación de las predicciones del nodo con los valores reales de la clase para los ejemplos de entrenamiento que se clasifican por el mismo nodo. Sin embargo, dado que el árbol se ha construido con estos ejemplos, el error puede infravalorarse, con lo que se compensa con el factor $(n+v)/(n-v)$, donde *n* es el número de ejemplos de entrenamiento que se clasifican por el nodo actual y *v* es el número de parámetros del modelo lineal. De esta forma, la estimación del error en un conjunto *I* de ejemplos se realizaría con la ecuación 2.30.

$$e(I) = \frac{n+v}{n-v} \times MAE = \frac{n+v}{n-v} \times \frac{\sum_{i \in I} |y_i - \hat{y}_i|}{n} \qquad \text{Ec. 2.30}$$

En la ecuación 2.30, *MAE* es el error medio absoluto [mean absolute error] del modelo, donde y_i es el valor de la clase para el ejemplo *i* y $\hat{y}_i$ la predicción del modelo para el mismo ejemplo. Para podar el árbol, se comienza por las hojas del mismo y se va comparando el error estimado para el nodo con el error estimado para los hijos del mismo, para lo cuál se emplea la ecuación 2.31.

$$e(subárbol) = \frac{e(i)|i| + e(d)|d|}{n} \qquad \text{Ec. 2.31}$$

En la ecuación 2.31, *e(i)* y *e(d)* son los errores estimados para los nodos hijo izquierdo y derecho, |*x*| el número de ejemplos que se clasifica por el nodo *x* y *n* el número de ejemplos que se clasifica por el nodo padre. Comparando el error estimado para el nodo con el error estimado para el subárbol, se decide podar si no es menor el error para el subárbol.

El proceso explicado hasta el momento sirve para el caso de que los atributos sean numéricos pero, si los atributos son nominales será preciso modificar el proceso: en primer lugar, se calcula el promedio de la clase en los ejemplos de entrenamiento para cada posible valor del atributo nominal, y se ordenan dichos valores de acuerdo a este promedio. Entonces, un atributo nominal con *k* posibles valores se transforma en *k-1* atributos binarios. El *i*-ésimo atributo binario tendrá, para un ejemplo dado, un *0* si el valor del atributo nominal es uno de los primeros *i* valores del orden establecido y un *1* en caso contrario. Con este proceso se logra tratar los atributos nominales como numéricos. También es necesario determinar cómo se actuará frente a los atributos para los que faltan valores. En este caso, se modifica ligeramente la ecuación 2.29 para llegar hasta la ecuación 2.32.

$$SDR = \frac{c}{|E|}\left[SD(E) - \sum_i \frac{|E_i|}{|E|} SD(E_i)\right] \qquad \text{Ec. 2.32}$$

En esta ecuación *c* es el número de ejemplos con el atributo conocido. Una vez explicadas las características de los árboles de predicción numérica, se pasa a mostrar el algoritmo *M5*, cuyo pseudocódigo se recoge en la figura 2.13.

```
M5 (ejemplos) {
   SD = sd(ejemplos)
   Para cada atributo nominal con k-valores
      convertir en k-1 atributos binarios
   raíz = nuevo nodo
   raíz.ejemplos = ejemplos
   Dividir(raíz)
   Podar(raíz)
   Dibujar(raíz)
}

Dividir(nodo) {
   Si tamaño(nodo.ejemplos)<4 O sd(nodo.ejemplos)<=0.05*SD Entonces
      nodo.tipo = HOJA
   Si no
      nodo.tipo = INTERIOR
      Para cada atributo
         Para cada posible punto de división del atributo
            calcular el SDR del atributo
      nodo.atributo = atributo con mayor SDR
      Dividir(nodo.izquierda)
      Dividir(nodo.derecha)
}

Podar(nodo) {
```

```
   Si nodo = INTERIOR
   Podar(nodo.hijoizquierdo)
   Podar(nodo.hijoderecho)
   nodo.modelo = RegresionLinear(nodo)
   Si ErrorSubarbol(nodo) > Error(nodo) Entonces
      nodo.tipo = HOJA
}

ErrorSubarbol(nodo) {
   l = nodo.izquierda
   r = nodo.derecha
   Si nodo = INTERIOR Entonces
      ErrorSubarbol = (tamaño(l.ejemplos)*ErrorSubarbol(l) +
tamaño(r.ejemplos)*ErrorSubarbol(r))tamaño(nodo.ejemplos)
   Si no
      ErrorSubarbol = error(nodo)
}
```

Figura 3.11: Pseudocódigo del algoritmo M5.

La función *RegresionLinear* generará la regresión correspondiente al nodo en el que nos encontramos. La función *error* evaluará el error del nodo mediante la ecuación 2.31.

3.4.3. Estimador de Núcleos

Los estimadores de densidad de núcleo [kernel density] son estimadores no paramétricos. De entre los que destaca el conocido histograma, por ser uno de los más antiguos y más utilizado, que tiene ciertas deficiencias relacionadas con la continuidad que llevaron a desarrollar otras técnicas. El estimador de núcleos fue propuesto por Rosenblatt en 1956 y Parzen en 1962 [DFL96]. La idea en la que se basan los estimadores de densidad de núcleo es la siguiente. Si X es una variable aleatoria con función de distribución F y densidad f, entonces en cada punto de continuidad x de f se confirma la ecuación 2.33.

$$f(x) = lim_{h \to 0} \frac{1}{2h}\left(F\left(x+h\right) - F\left(x-h\right)\right) \quad \text{Ec. 2.33}$$

Dada una muestra $X_1,...,X_n$ proveniente de la distribución F, para cada h fijo, $F(x+h)-F(x-h)$ se puede estimar por la proporción de observaciones que están dentro del intervalo $(x-h, x+h)$. Por lo tanto, tomando h pequeño, un estimador natural de la densidad es el que se muestra en la ecuación 2.34, donde $\#A$ es el número de elementos del conjunto A.

$$\hat{f}_{n,h}(x) = \frac{1}{2hn}\#\left\{X_i : X_i \in \left(x-h, x+h\right)\right\} \quad \text{Ec. 2.34}$$

Otra manera de expresar este estimador es considerando la función de peso w definida como se muestra en la ecuación 2.35, de manera que el estimador de la densidad f en el punto x se puede expresar como se expresa en la ecuación 2.36.

$$w(x) = \begin{cases} 1/2 & si\ |x| < 1 \\ 0 & en\ c.c. \end{cases}$$ Ec. 2.35

$$\hat{f}_{n,h}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}w\left(\frac{x - X_i}{h}\right)$$ Ec. 2.36

Pero este estimador no es una función continua, ya que tiene saltos en los puntos $X_i \pm h$ y su derivada es *0* en todos los otros puntos. Por ello se ha sugerido reemplazar a la función *w* por funciones más suaves *K*, llamadas núcleos, lo que da origen a los estimadores de núcleos. El estimador de núcleos de una función de densidad *f* calculado a partir de una muestra aleatoria $X_1,...,X_n$ de dicha densidad se define según la ecuación 2.37.

$$\hat{f}_{n,h}(x) = \frac{1}{nh}\sum_{i=1}^{n}K\left(\frac{x - X_i}{h}\right)$$ Ec. 2.37

En la ecuación 2.37, la función *K* se elige generalmente entre las funciones de densidad conocidas, por ejemplo gaussiana, que se muestra en la ecuación 2.38, donde σ es la desviación típica de la distribución y μ la media.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$ Ec. 2.38

El otro parámetro de la ecuación 2.37 es *h*, llamado ventana, parámetro de suavizado o ancho de banda, el cual determina las propiedades estadísticas del estimador: el sesgo crece y la varianza decrece con *h* [HALI94]. Es decir que si *h* es grande, los estimadores están sobresuavizados y son sesgados, y si *h* es pequeño, los estimadores resultantes están subsuavizados, lo que equivale a decir que su varianza es grande.

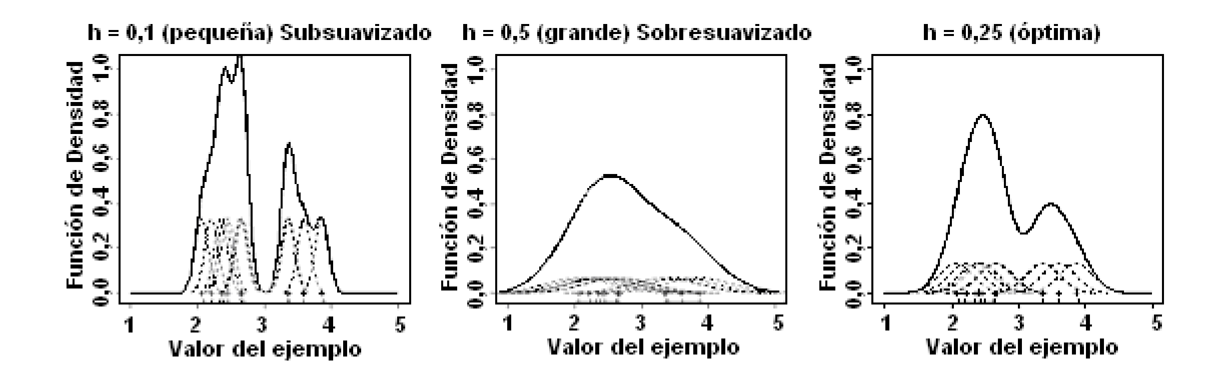


Figura 3.12: Importancia del parámetro "tamaño de ventana" en el estimador de núcleos.

A pesar de que la elección del núcleo *K* determina la forma de la densidad estimada, la literatura sugiere que esta elección no es crítica, al menos entre las alternativas usuales [DEA97]. Más importante es la elección del tamaño de ventana. En la figura 2.14 se muestra cómo un valor pequeño para este factor hace que la

función de distribución generada esté subsuavizada. Mientras, al emplear un *h* demasiado grande provoca el sobresuavizado de la función de distribución. Por último, empleando el *h* óptimo se obtiene la función de distribución adecuada.

Para determinar un ancho de banda con el cual comenzar, una alternativa es calcular el ancho de banda óptimo si se supone que la densidad tiene una forma específica. La ventana óptima en el sentido de minimizar el error medio cuadrático integrado, definido como la esperanza de la integral del error cuadrático sobre toda la densidad, fue calculada por Bowman [BOW85], y Silverman [SIL86] y depende de la verdadera densidad *f* y del núcleo *K*. Al suponer que ambos, la densidad y el núcleo son normales, la ventana óptima resulta ser la que se muestra en la ecuación 2.39.

$$h^* = 1.06\,\sigma\, n^{-1/5} \qquad \text{Ec. 2.39}$$

En la ecuación 2.39 σ es la desviación típica de la densidad. La utilización de esta *h* será adecuada si la población se asemeja en su distribución a la de la normal; sin embargo si trabajamos con poblaciones multimodales se producirá una sobresuavización de la estimación. Por ello el mismo autor sugiere utilizar medidas robustas de dispersión en lugar de σ, con lo cual el ancho de banda óptimo se obtiene como se muestra en la ecuación 2.40.

$$h^* = 1.06\, min(\sigma, 0.75\, IQR)\, n^{-1/5} \qquad \text{Ec. 2.40}$$

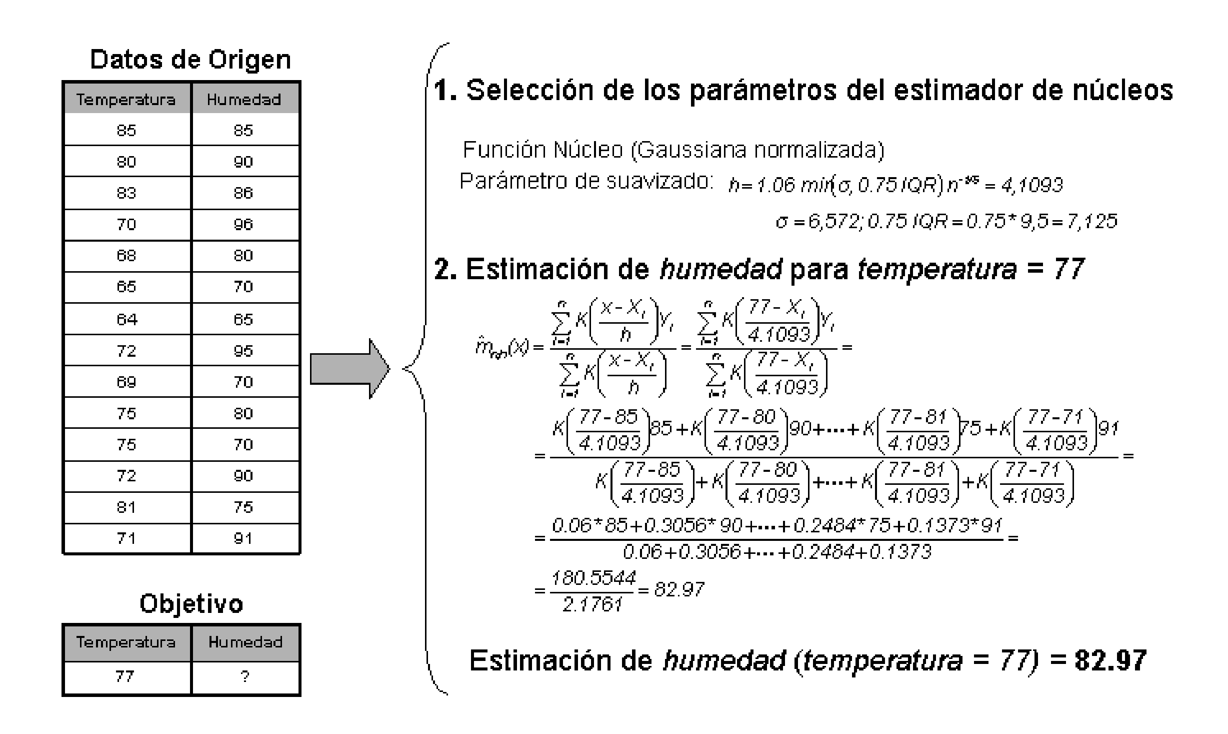
En la ecuación 2.40 *IQR* es el rango intercuartílico, esto es, la diferencia entre los percentiles *75* y *25* [DEA97].

Una vez definidos todos los parámetros a tener en cuenta para emplear un estimador de núcleos, hay que definir cómo se obtiene, a partir del mismo, el valor de la variable a predecir, *y*, en función del valor de la variable dependiente, *x*. Esto se realiza mediante el estimador de Nadaraya-Watson, que se muestra en la ecuación 2.41.

$$\hat{m}(x) = E(Y \mid X = x) = \frac{\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) Y_i}{\sum_{r=1}^{n} K\left(\frac{x - X_r}{h}\right)} \qquad \text{Ec. 2.41}$$

En la ecuación 2.41 *x* es el valor del atributo dependiente a partir del cual se debe obtener el valor de la variable independiente *y*; Y_i es el valor del atributo independiente para el ejemplo de entrenamiento *i*.

Una vez completada la explicación de cómo aplicar los estimadores de núcleos para predecir el valor de una clase numérica, se muestra, en la figura 2.15, un ejemplo de su utilización basado en los ejemplos de la tabla 2.1 (apartado 2.5), tomando la variable *temperatura* como predictora y la variable *humedad* como dependiente o a predecir.


Datos de Origen

Temperatura	Humedad
85	85
80	90
83	86
70	96
68	80
65	70
64	65
72	95
69	70
75	80
75	70
72	90
81	75
71	91

Objetivo

Temperatura	Humedad
77	?

Figura 3.13: Ejemplo de predicción con un estimador de núcleos.

En primer lugar se definen los parámetros que se van a emplear para el estimador de núcleos: la función núcleo y el parámetro de suavizado. Posteriormente se puede realizar la predicción, que en este caso consiste en predecir el valor del atributo *humedad* sabiendo que la *temperatura* es igual a *77*. Después de completar el proceso se determina que el valor de la humedad es igual a *82.97*.

Aplicación a problemas multivariantes

Hasta el momento se han explicado las bases sobre las que se sustentan los estimadores de núcleos, pero en los problemas reales no es una única variable la que debe tratarse, sino que han de tenerse en cuenta un número indeterminado de variables. Por ello, es necesario ampliar el modelo explicado para permitir la introducción de d variables. Así, supongamos n ejemplos X_i, siendo X_i un vector d-dimensional. El estimador de núcleos de la función de densidad f calculado a partir de la muestra aleatoria $X_1,...,X_n$ de dicha densidad se define como se muestra en la ecuación 2.42.

$$\hat{f}_{n,H}(x) = \frac{1}{n|H|} \sum_{i=1}^{n} K\left(H^{-1}(x - X_i)\right) \qquad \text{Ec. 2.42}$$

Tal y como puede verse, la ecuación 2.42 es una mera ampliación de la ecuación 2.37: en este caso H no es ya un único valor numérico, sino una matriz simétrica y definida positiva de orden $d \times d$, denominada matriz de anchos de ventana. Por su parte K es generalmente una función de densidad multivariante. Por ejemplo, la función gaussiana normalizada en este caso pasaría a ser la que se muestra en la ecuación 2.43.

$$f(x) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{x^T x}{2}} \quad \text{Ec. 2.43}$$

De nuevo, es más importante definir una correcta matriz *H* que la función núcleo elegida. También el estimador de Nadaraya-Watson, que se muestra en la ecuación 2.44, es una ampliación del visto en la ecuación 2.41.

$$\hat{m}(x) = E(Y \mid X = x) = \frac{\sum_{i=1}^{n} K\left(H^{-1}(x - X_i)\right) Y_i}{\sum_{r=1}^{n} K\left(H^{-1}(x - X_i)\right)} \quad \text{Ec. 2.44}$$

Tal y como se ve en la ecuación 2.44, el cambio radica en que se tiene una matriz de anchos de ventana en lugar de un único valor de ancho de ventana.

Aplicación a problemas de clasificación

Si bien los estimadores de núcleo son diseñados para la predicción numérica, también pueden utilizarse para la clasificación. En este caso, se dispone de un conjunto de *c* clases a las que puede pertenecer un ejemplo determinado. Y estos ejemplos se componen de *d* variables o atributos. Se puede estimar la densidad de la clase *j* mediante la ecuación 2.45, en la que n_j es el número de ejemplos de entrenamiento que pertenecen a la clase *j*, Y_i^j será *1* en caso de que el ejemplo *i* pertenezca a la clase *j* y *0* en otro caso, *K* vuelve a ser la función núcleo y *h* el ancho de ventana. En este caso se ha realizado la simplificación del modelo multivariante, empleando en lugar de una matriz de anchos de ventana un único valor escalar porque es el modelo que se utiliza en la implementación que realiza WEKA de los estimadores de núcleo.

$$\hat{f}_j(x) = \frac{1}{n_j} \sum_{i=1}^{n} Y_i^j h^{-d} K\left(\frac{x - X_i}{h}\right) \quad \text{Ec. 2.45}$$

La probabilidad *a priori* de que un ejemplo pertenezca a la clase *j* es igual a $P_j = n_j / n$. Se puede estimar la probabilidad *a posteriori*, definida mediante $q_j(x)$, de que el ejemplo pertenezca a *j*, tal y como se muestra en la ecuación 2.46.

$$q_j(x) = \frac{P_j \hat{f}_j(x)}{f(x)} \approx \frac{P_j \hat{f}_j(x)}{\sum_{k=1}^{c} P_k \hat{f}_k(x)} = \frac{\sum_{i=1}^{n} Y_i^j h^{-d} K\left(\frac{x - X_i}{h}\right)}{\sum_{r=1}^{n} h^{-d} K\left(\frac{x - X_r}{h}\right)} = \hat{q}_j(x) \quad \text{Ec. 2.46}$$

De esta forma, el estimador en este caso es idéntico al estimador de Nadayara-Watson representado en las ecuaciones 2.41 y 2.44.

Por último, se muestra un ejemplo de la aplicación de un estimador de núcleos a un problema de clasificación: se trata del problema planteado en la tabla 2.1 (apartado 2.5), y más concretamente se trata de predecir el valor de la clase *jugar* a partir únicamente del atributo numérico *temperatura*. Este ejemplo se muestra en la figura 2.16.

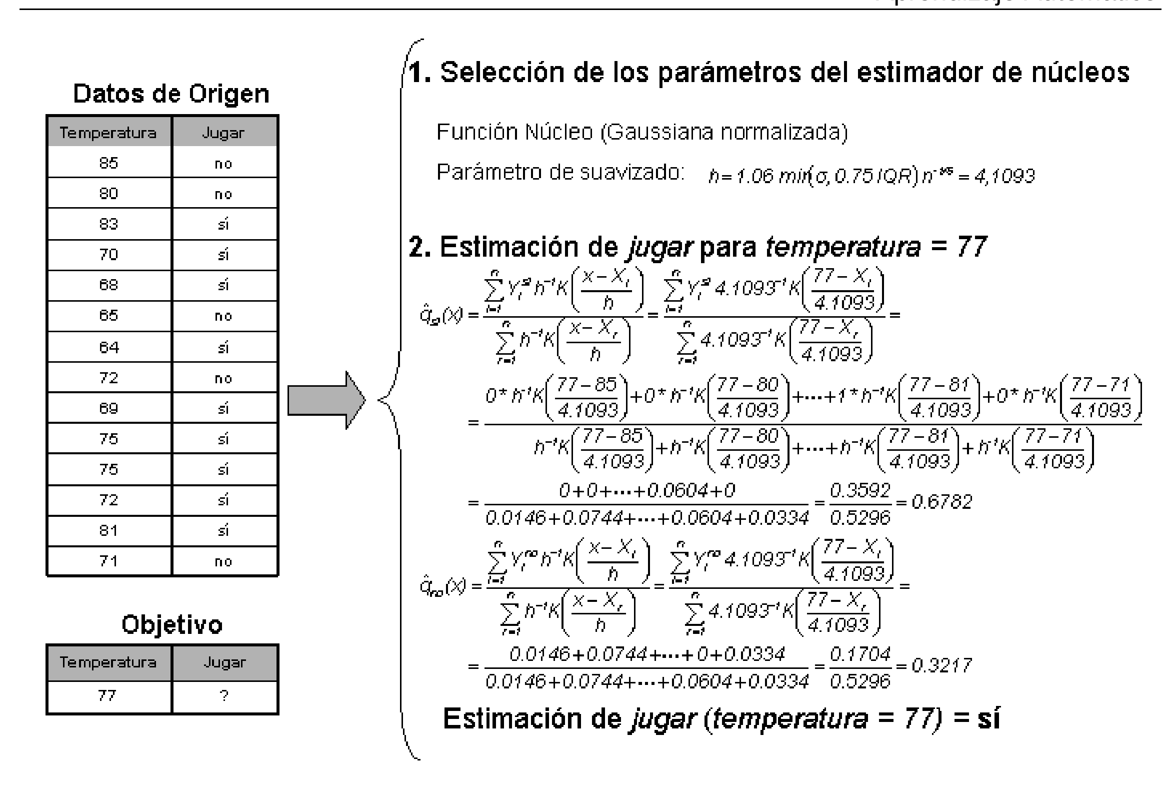


Figura 3.14: Ejemplo de clasificación mediante un estimador de núcleos.

Al igual que para el problema de predicción, en primer lugar se definen los parámetros del estimador de núcleos para, posteriormente, estimar la clase a la que pertenece el ejemplo de test. En este caso se trata de predecir si se puede jugar o no al tenis teniendo en cuenta que la *temperatura* es igual a *77*. Y la conclusión a la que se llega utilizando el estimador de núcleos es que sí se puede jugar.

3.5. La clasificación

La clasificación es el proceso de dividir un conjunto de datos en grupos mutuamente excluyentes [WK91, LAN96, MIT97], de tal forma que cada miembro de un grupo esté lo mas cerca posible de otros y grupos diferentes estén lo más lejos posible de otros, donde la distancia se mide con respecto a las variables especificadas, que se quieren predecir.

Tabla2.1. Ejemplo de problema de clasificación.

Ejemplo	*Vista*	*Temperatura*	*Humedad*	*Viento*	*Jugar*
1	Soleado	Alta (85)	Alta (85)	No	No
2	Soleado	Alta (80)	Alta (90)	Sí	No
3	Nublado	Alta (83)	Alta (86)	No	Sí
4	Lluvioso	Media (70)	Alta (96)	No	Sí

5	Lluvioso	Baja (68)	Normal (80)	No	Sí
6	Lluvioso	Baja (65)	Normal (70)	Sí	No
7	Nublado	Baja (64)	Normal (65)	Sí	Sí
8	Soleado	Media (72)	Alta (95)	No	No
9	Soleado	Baja (69)	Normal (70)	No	Sí
10	Lluvioso	Media (75)	Normal (80)	No	Sí
11	Soleado	Media (75)	Normal (70)	Sí	Sí
12	Nublado	Media (72)	Alta (90)	Sí	Sí
13	Nublado	Alta (81)	Normal (75)	No	Sí
14	Lluvioso	Media (71)	Alta (91)	Sí	No

El ejemplo empleado tiene dos atributos, temperatura y humedad, que pueden emplearse como simbólicos o numéricos. Entre paréntesis se presentan sus valores numéricos.

En los siguientes apartados se presentan y explican las principales técnicas de clasificación. Además, se mostrarán ejemplos que permiten observar el funcionamiento del algoritmo, para lo que se utilizará la tabla 2.1, que presenta un sencillo problema de clasificación consistente en, a partir de los atributos que modelan el tiempo (vista, temperatura, humedad y viento), determinar si se puede o no jugar al tenis.

3.5.1. Tabla de Decisión

La tabla de decisión constituye la forma más simple y rudimentaria de representar la salida de un algoritmo de aprendizaje, que es justamente representarlo como la entrada.

Estos algoritmos consisten en seleccionar subconjuntos de atributos y calcular su precisión [accuracy] para predecir o clasificar los ejemplos. Una vez seleccionado el mejor de los subconjuntos, la tabla de decisión estará formada por los atributos seleccionados (más la clase), en la que se insertarán todos los ejemplos de entrenamiento únicamente con el subconjunto de atributos elegido. Si hay dos ejemplos con exactamente los mismos pares *atributo-valor* para todos los atributos del subconjunto, la clase que se elija será la media de los ejemplos (en el caso de una clase numérica) o la que mayor probabilidad de aparición tenga (en el caso de una clase simbólica).

La precisión de un subconjunto *S* de atributos para todos los ejemplos de entrenamientos se calculará mediante la ecuación 2.47 para el caso de que la clase sea simbólica o mediante la ecuación 2.48 en el caso de que la clase sea numérica:

$$precisión(S) = \frac{ejemplos\ bien\ clasificados}{ejemplos\ totales} \qquad \text{Ec. 2.47}$$

$$precisión(S) = -RMSE = -\sqrt{\frac{\sum_{i \in I} (y_i - \hat{y}_i)^2}{n}} \qquad \text{Ec. 2.48}$$

Donde, en la ecuación 2.48, RMSE es la raíz cuadrada del error cuadrático medio [root mean squared error], *n* es el número de ejemplos totales, y_i el valor de la clase para el ejemplo *i* y $\hat{y}_i$ el valor predicho por el modelo para el ejemplo *i*.

Como ejemplo de tabla de decisión, simplemente se puede utilizar la propia tabla 2.1, dado que si se comenzase a combinar atributos y a probar la precisión de dicha combinación, se obtendría como resultado que los cuatro atributos deben emplearse, con lo que la tabla de salida sería la misma. Esto no tiene por qué ser así, ya que en otros problemas no serán necesarios todos los atributos para generar la tabla de decisión, como ocurre en el ejemplo de la tabla 2.2 donde se dispone de un conjunto de entrenamiento en el que aparecen los atributos sexo, y tipo (tipo de profesor) y la clase a determinar es si el tipo de contrato es o no fijo.

Tabla2.2. Determinación del tipo de contrato.

	Atributos		***Clase***
Ejemplo Nº	***Sexo***	***Tipo***	***Fijo***
1	Hombre	Asociado	No
2	Mujer	Catedrático	Si
3	Hombre	Titular	Si
4	Mujer	Asociado	No
5	Hombre	Catedrático	Si
6	Mujer	Asociado	No
7	Hombre	Ayudante	No
8	Mujer	Titular	Si

9	Hombre	Asociado	No
10	Mujer	Ayudante	No
11	Hombre	Asociado	No

Si se toma como primer subconjunto el formado por el atributo sexo, y se eliminan las repeticiones resulta la tabla 2.3

Tabla2.3. Subconjunto 1.

Ejemplo Nº	*Sexo*	*Fijo*
1	Hombre	No
2	Mujer	Si
3	Hombre	Si
4	Mujer	No

Con lo que se pone de manifiesto que la probabilidad de clasificar bien es del 50%. Si por el contrario se elimina el atributo *Sexo,* quedará la tabla 2.4.

Tabla2.4. Subconjunto 2.

Ejemplo Nº	*Tipo*	*Fijo*
1	Asociado	No
2	Catedrático	Si
3	Titular	Si
7	Ayudante	No

Que tiene una precisión de aciertos del 100%, por lo que se deduce que ésta última tabla es la que se debe tomar como tabla de decisión. El resultado es lógico ya que el atributo sexo es irrelevante a la hora de determinar si el contrato es o no fijo.

3.5.2. Árboles de Decisión

El aprendizaje de árboles de decisión está englobado como una metodología del aprendizaje supervisado. La representación que se utiliza para las descripciones del concepto adquirido es el árbol de decisión, que consiste en una representación del conocimiento relativamente simple y que es una de las causas por la que los procedimientos utilizados en su aprendizaje son más sencillos que los de sistemas que

utilizan lenguajes de representación más potentes, como redes semánticas, representaciones en lógica de primer orden etc. No obstante, la potencia expresiva de los árboles de decisión es también menor que la de esos otros sistemas. El aprendizaje de árboles de decisión suele ser más robusto frente al ruido y conceptualmente sencillo, aunque los sistemas que han resultado del perfeccionamiento y de la evolución de los más antiguos se complican con los procesos que incorporan para ganar fiabilidad. La mayoría de los sistemas de aprendizaje de árboles suelen ser no incrementales, pero existe alguna excepción [UTG88].

El primer sistema que construía árboles de decisión fue CLS de Hunt, desarrollado en 1959 y depurado a lo largo de los años sesenta. CLS es un sistema desarrollado por psicólogos como un modelo del proceso cognitivo de formación de conceptos sencillos. Su contribución fundamental fue la propia metodología pero no resultaba computacionalmente eficiente debido al método que empleaba en la extensión de los nodos. Se guiaba por una estrategia similar al *minimax* con una función que integraba diferentes costes.

En 1979 Quinlan desarrolla el sistema ID3 [QUIN79], que él denominaría simplemente herramienta porque la consideraba experimental. Conceptualmente es fiel a la metodología de CLS pero le aventaja en el método de expansión de los nodos, basado en una función que utiliza la medida de la información de Shannon. La versión definitiva, presentada por su autor Quinlan como un sistema de aprendizaje, es el sistema C4.5 que expone con cierto detalle en la obra *C4.5: Programs for Machine Learning* [QUIN93]. La evolución -comercial- de ese sistema es otro denominado C5 del mismo autor, del que se puede obtener una versión de demostración restringida en cuanto a capacidades; por ejemplo, el número máximo de ejemplos de entrenamiento.

Representación de un árbol de decisión

Un árbol de decisión [MUR98] puede interpretarse esencialmente como una serie de *reglas* compactadas para su representación en forma de árbol. Dado un conjunto de ejemplos, estructurados como vectores de pares ordenados atributo-valor, de acuerdo con el formato general en el aprendizaje inductivo a partir de ejemplos, el concepto que estos sistemas adquieren durante el proceso de aprendizaje consiste en un árbol. Cada eje está etiquetado con un par atributo-valor y las hojas con una clase, de forma que la trayectoria que determinan desde la raíz los pares de un ejemplo de entrenamiento alcanzan una hoja etiquetada -normalmente- con la clase del ejemplo. La clasificación de un ejemplo nuevo del que se desconoce su clase se hace con la misma técnica, solamente que en ese caso al atributo clase, cuyo valor se desconoce, se le asigna de acuerdo con la etiqueta de la hoja a la que se accede con ese ejemplo.

Problemas apropiados para este tipo de aprendizaje

Las características de los problemas apropiados para resolver mediante este aprendizaje dependen del sistema de aprendizaje específico utilizado, pero hay una serie de ellas generales y comunes a la mayoría y que se describen a continuación:

- Que la representación de los ejemplos sea mediante vectores de pares atributo-valor, especialmente cuando los valores son disjuntos y en un número pequeño. Los sistemas actuales están preparados para tratar atributos con valores continuos, valores desconocidos e incluso valores con una distribución de probabilidad.

- Que el atributo que hace el papel de la clase sea de tipo discreto y con un número pequeño de valores, sin embargo existen sistemas que adquieren como concepto aprendido funciones con valores continuos.

- Que las descripciones del concepto adquirido deban ser expresadas en forma normal disyuntiva.

- Que posiblemente existan errores de clasificación en el conjunto de ejemplos de entrenamiento, así como valores desconocidos en algunos de los atributos en algunos ejemplos Estos sistemas, por lo general, son robustos frente a los errores del tipo mencionado.

A continuación se presentan tres algoritmos de árboles de decisión, los dos primeros diseñados por Quinlan [QUIN86, QUIN93], los sistemas ID3 y C4.5; y el tercero un árbol de decisión muy sencillo, con un único nivel de decisión.

- **El sistema ID3**

El sistema ID3 [QUIN86] es un algoritmo simple y, sin embargo, potente, cuya misión es la elaboración de un árbol de decisión. El procedimiento para generar un árbol de decisión consiste, como se comentó anteriormente en seleccionar un atributo como raíz del árbol y crear una rama con cada uno de los posibles valores de dicho atributo. Con cada rama resultante (nuevo nodo del árbol), se realiza el mismo proceso, esto es, se selecciona otro atributo y se genera una nueva rama para cada posible valor del atributo. Este procedimiento continúa hasta que los ejemplos se clasifiquen a través de uno de los caminos del árbol. El nodo final de cada camino será un nodo hoja, al que se le asignará la clase correspondiente. Así, el objetivo de los árboles de decisión es obtener reglas o relaciones que permitan clasificar a partir de los atributos.

En cada nodo del árbol de decisión se debe seleccionar un atributo para seguir dividiendo, y el criterio que se toma para elegirlo es: se selecciona el atributo que mejor separe (ordene) los ejemplos de acuerdo a las clases. Para ello se emplea la entropía, que es una medida de cómo está ordenado el universo. La teoría de la información (basada en la entropía) calcula el número de bits (información, preguntas sobre atributos) que hace falta suministrar para conocer la clase a la que pertenece un ejemplo. Cuanto menor sea el valor de la entropía, menor será la incertidumbre y más útil será el atributo para la clasificación. La definición de entropía que da Shannon en su *Teoría de la Información* (1948) es: Dado un conjunto de eventos $A=\{A_1, A_2,..., A_n\}$, con probabilidades $\{p_1, p_2,..., p_n\}$, la información en el conocimiento de un suceso A_i

(bits) se define en la ecuación 2.49, mientras que la información media de *A* (bits) se muestra en la ecuación 2.50.

$$I(A_i) = log_2\left(\frac{1}{p_i}\right) = -log_2(p_i) \quad \text{Ec. 2.49}$$

$$I(A) = \sum_{i=1}^{n} p_i I(A_i) = -\sum_{i=1}^{n} p_i log_2(p_i) \quad \text{Ec. 2.50}$$

Si aplicamos la entropía a los problemas de clasificación se puede medir lo que se discrimina (se gana por usar) un atributo A_i empleando para ello la ecuación 2.51, en la que se define la ganancia de información.

$$G(A_i) = I - I(A_i) \quad \text{Ec. 2.51}$$

Siendo *I* la información antes de utilizar el atributo e $I(A_i)$ la información después de utilizarlo. Se definen ambas en las ecuaciones 2.52 y 2.53.

$$I = -\sum_{c=1}^{nc} \frac{n_c}{n} log_2\left(\frac{n_c}{n}\right) \quad \text{Ec. 2.52}$$

$$I(A_i) = \sum_{j=1}^{nv(A_i)} \frac{n_{ij}}{n} I_{ij} \; ; \; I_{ij} = -\sum_{k=1}^{nc} \frac{n_{ijk}}{n_{ij}} log_2\left(\frac{n_{ijk}}{n_{ij}}\right) \quad \text{Ec. 2.53}$$

En estas ecuaciones *nc* será el número de clases y n_c el número de ejemplares de la clase *c*, siendo *n* el número total de ejemplos. Será $nv(A_i)$ el número de valores del atributo A_i, n_{ij} el número de ejemplos con el valor *j* en A_i y n_{ijk} el número de ejemplos con valor *j* en A_i y que pertenecen a la clase *k*. Una vez explicada la heurística empleada para seleccionar el mejor atributo en un nodo del árbol de decisión, se muestra el algoritmo ID3:

```
1. Seleccionar el atributo Ai que maximice la ganancia G(Ai).

2. Crear un nodo para ese atributo con tantos sucesores como
   valores tenga.

3. Introducir los ejemplos en los sucesores según el valor que
   tenga el atributo Ai.

4. Por cada sucesor:

   a. Si sólo hay ejemplos de una clase, Ck, entonces etiquetarlo
      con Ck.

   b. Si no, llamar a ID3 con una tabla formada por los ejemplos
      de ese nodo, eliminando la columna del atributo Ai.
```

Figura 3.15: Pseudocódigo del algoritmo ID3.

Por último, en la figura 2.18 se representa el proceso de generación del árbol de decisión para el problema planteado en la tabla 2.1.

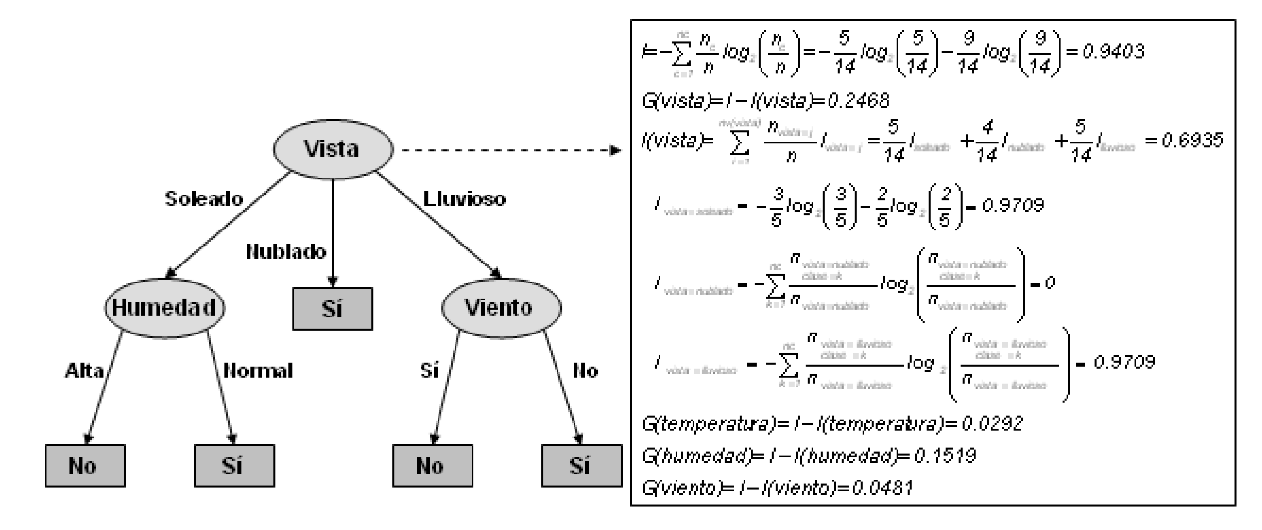


Figura 3.16: Ejemplo de clasificación con ID3.

En la figura 2.18 se muestra el árbol de decisión que se generaría con el algoritmo ID3. Además, para el primer nodo del árbol se muestra cómo se llega a decidir que el mejor atributo para dicho nodo es *vista*. Se generan nodos para cada valor del atributo y, en el caso de *vista* = *Nublado* se llega a un nodo *hoja* ya que todos los ejemplos de entrenamiento que llegan a dicho nodo son de clase *Sí*. Sin embargo, para los otros dos casos se repite el proceso de elección con el resto de atributos y con los ejemplos de entrenamiento que se clasifican a través de ese nodo.

- **El sistema C4.5**

El ID3 es capaz de tratar con atributos cuyos valores sean discretos o continuos. En el primer caso, el árbol de decisión generado tendrá tantas *ramas* como valores posibles tome el atributo. Si los valores del atributo son continuos, el ID3 no clasifica correctamente los ejemplos dados. Por ello, Quinlan [QUIN93] propuso el C4.5, como extensión del ID3, que permite:

1. Empleo del concepto razón de ganancia (GR, [Gain Ratio])
2. Construir árboles de decisión cuando algunos de los ejemplos presentan valores desconocidos para algunos de los atributos.
3. Trabajar con atributos que presenten valores continuos.
4. La *poda* de los árboles de decisión [QUIN87, QR89].

5. Obtención de Reglas de Clasificación.

Razón de Ganancia

El test basado en el criterio de maximizar la ganancia tiene como sesgo la elección de atributos con muchos valores. Esto es debido a que cuanto más fina sea la participación producida por los valores del atributo, normalmente, la incertidumbre o entropía en cada nuevo nodo será menor, y por lo tanto también será menor la media de la entropía a ese nivel. C4.5 modifica el criterio de selección del atributo empleando en lugar de la *ganancia* la *razón de ganancia*, cuya definición se muestra en la ecuación 2.54.

$$GR(A_i) = \frac{G(A_i)}{I(División\ A_i)} = \frac{G(A_i)}{-\sum_{j=1}^{nv(A_i)} \frac{n_{ij}}{n} log_2\left(\frac{n_{ij}}{n}\right)} \qquad \text{Ec. 2.54}$$

Al término *I(División* A_i*)* se le denomina información de ruptura. En esta medida cuando n_{ij} tiende a *n*, el denominador se hace *0*. Esto es un problema aunque según Quinlan, la razón de ganancia elimina el sesgo.

Valores Desconocidos

El sistema C4.5 admite ejemplos con atributos desconocidos tanto en el proceso de aprendizaje como en el de validación. Para calcular, durante el proceso de aprendizaje, la razón de ganancia de un atributo con valores desconocidos, se redefinen sus dos términos, la ganancia, ecuación 2.55, y la información de ruptura, ecuación 2.56.

$$G(A_i) = \frac{n_{ic}}{n}(I - I(A_i)) \qquad \text{Ec. 2.55}$$

$$I(División\ A_i) = -\left(\sum_{j=1}^{nv(A_i)} \frac{n_{ij}}{n} log_2\left(\frac{n_{ij}}{n}\right)\right) - \frac{n_{id}}{n} log_2\left(\frac{n_{id}}{n}\right) \qquad \text{Ec. 2.56}$$

En estas ecuaciones, n_{ic} es el número de ejemplos con el atributo *i* conocido, y n_{id} el número de ejemplos con valor desconocido en el mismo atributo. Además, para el cálculo de las entropía *I(*A_i*)* se tendrán en cuenta únicamente los ejemplos en los que el atributo A_i tenga un valor definido.

No se toma el valor desconocido como significativo, sino que se supone una distribución probabilística del atributo de acuerdo con los valores de los ejemplos en la muestra de entrenamiento. Cuando se entrena, los casos con valores desconocidos se distribuyen con pesos de acuerdo a la frecuencia de aparición de cada posible valor del atributo en el resto de ejemplos de entrenamiento. El peso ω_{ij} con que un ejemplo *i* se distribuiría desde un nodo etiquetado con el atributo *A* hacia el hijo con valor *j* en

dicho atributo se calcula mediante la ecuación 2.57, en la que ω_i es el peso del ejemplo i al llegar al nodo, esto es, antes de distribuirse, y *p(A=j)* la suma de pesos de todos los ejemplos del nodo con valor *j* en el atributo *A* entre la suma total de pesos de todos los ejemplos del nodo (ω).

$$\omega_{ij} = \omega_i \, p(A = j) = \omega_i \frac{\omega_{A=j}}{\omega} \qquad \text{Ec. 2.57}$$

En cuanto a la clasificación de un ejemplo de test, si se alcanza un nodo con un atributo que el ejemplo no tiene (desconocido), se distribuye el ejemplo (divide) en tantos casos como valores tenga el atributo, y se da un peso a cada resultado con el mismo criterio que en el caso del entrenamiento: la frecuencia de aparición de cada posible valor del atributo en los ejemplos de entrenamiento. El resultado de esta técnica es una clasificación con probabilidades, correspondientes a la distribución de ejemplos en cada nodo hoja.

Atributos Continuos

El tratamiento que realiza C4.5 de los atributos continuos está basado en la ganancia de información, al igual que ocurre con los atributos discretos. Si un atributo continuo A_i presenta los valores ordenados $v_1, v_2, \ldots, v_n$, se comprueba cuál de los valores $z_i = (v_i + v_{i+1})/2$; $1 \leq j < n$, supone una ruptura del intervalo $[v_1, v_n]$ en dos subintervalos $[v_1, z_j]$ y $(z_j, v_n]$ con mayor ganancia de información. El atributo continuo, ahora con dos únicos valores posibles, entrará en competencia con el resto de los atributos disponibles para expandir el nodo.

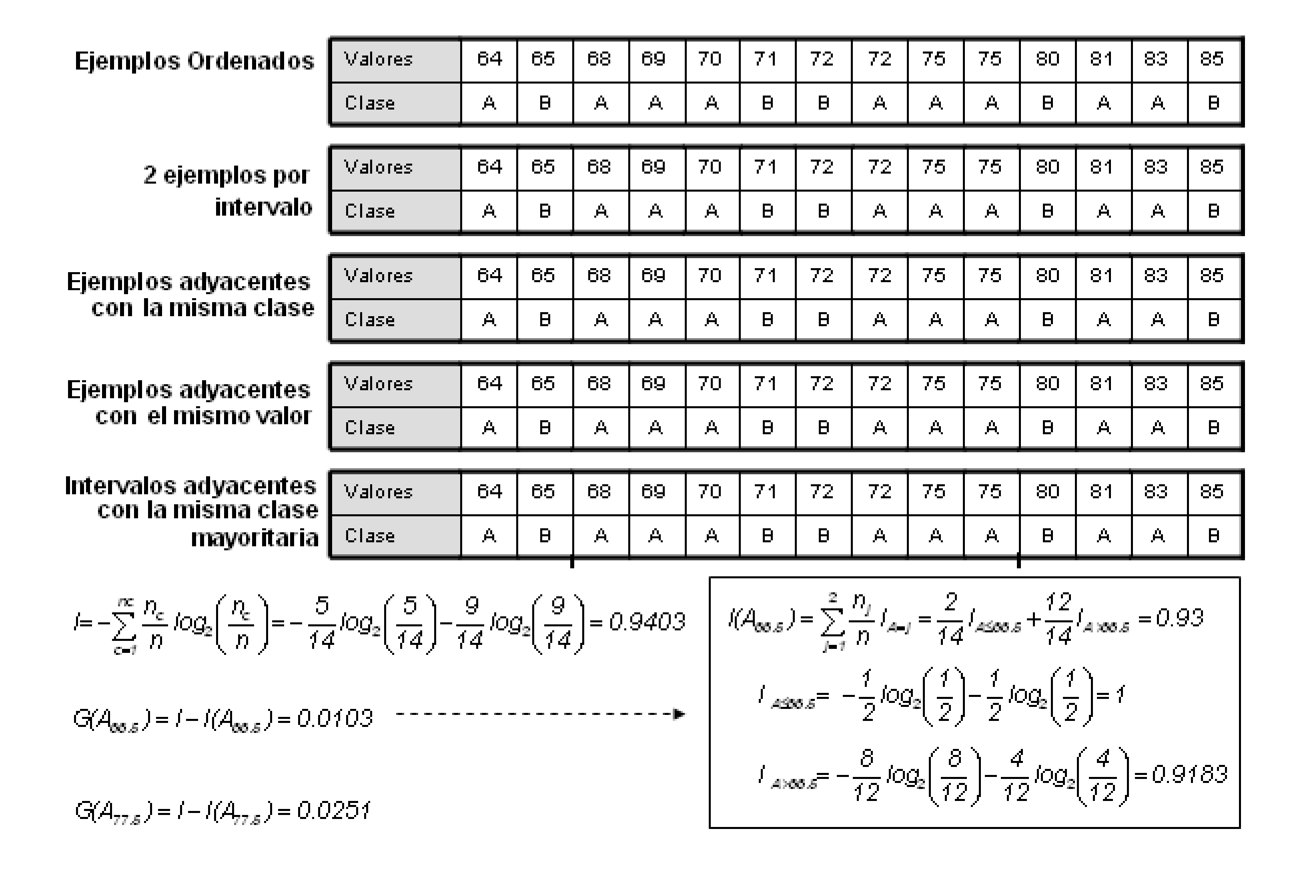


Figura 3.17: Ejemplo de tratamiento de atributos continuos con C4.5.

Para mejorar la eficiencia del algoritmo no se consideran todos los posibles puntos de corte, sino que se tienen en cuenta las siguientes reglas:

1. Cada subintervalo debe tener un número mínimo de ejemplos (por ejemplo, 2).
2. No se divide el intervalo si el siguiente ejemplo pertenece a la misma clase que el actual.
3. No se divide el intervalo si el siguiente ejemplo tiene el mismo valor que el actual.
4. Se unen subintervalos adyacentes si tienen la misma clase mayoritaria.

Como se ve en el ejemplo de la figura 2.19, aplicando las reglas anteriores sólo es preciso probar dos puntos de corte (*66,5* y *77,5*), mientras que si no se empleara ninguna de las mejoras que se comentaron anteriormente se deberían haber probado un total de trece puntos. Como se ve en la figura 2.19, finalmente se tomaría como punto de ruptura el *77,5*, dado que obtiene una mejor ganancia. Una vez seleccionado el punto de corte, este atributo numérico competiría con el resto de atributos. Si bien aquí se ha empleado la ganancia, realmente se emplearía la razón de ganancia, pero no afecta a la elección del punto de corte. Cabe mencionar que ese atributo no deja de estar disponible en niveles inferiores como en el caso de los discretos, aunque con sus valores restringidos al intervalo que domina el camino.

Poda del árbol de decisión

El árbol de decisión ha sido construido a partir de un conjunto de ejemplos, por tanto, reflejará correctamente todo el grupo de casos. Sin embargo, como esos ejemplos pueden ser muy diferentes entre sí, el árbol resultante puede llegar a ser bastante complejo, con trayectorias largas y muy desiguales. Para facilitar la comprensión del árbol puede realizarse una *poda* del mismo. C4.5 efectúa la poda después de haber desarrollado el árbol completo (*post-poda*), a diferencia de otros sistemas que realizan la construcción del árbol y la poda a la vez (pre-poda); es decir, estiman la necesidad de seguir desarrollando un nodo aunque no posea el carácter de hoja. En C4.5 el proceso de podado comienza en los nodos hoja y recursivamente continúa hasta llegar al nodo raíz. Se consideran dos operaciones de poda en C4.5: reemplazo de sub-árbol por hoja (*subtree replacement*) y elevación de sub-árbol (*subtree raising*). En la figura 2.20 se muestra en lo que consiste cada tipo de poda.

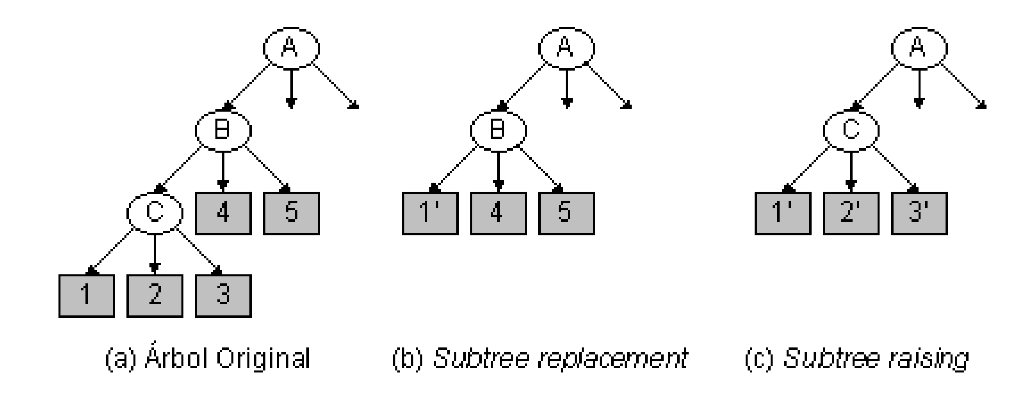


Figura 3.18: Tipos de operaciones de poda en C4.5.

En esta figura tenemos el árbol original antes del podado *(a)*, y las dos posibles acciones de podado a realizar sobre el nodo interno *C*. En *(b)* se realiza *subtree replacement*, en cuyo caso el nodo *C* es reemplazado por uno de sus subárboles. Por último, en *(c)* se realiza *subtree raising*: El nodo *B* es sustituido por el subárbol con raíz *C*. En este último caso hay que tener en cuenta que habrá que reclasificar de nuevo los ejemplos a partir del nodo *C*. Además, *subtree raising* es muy costoso computacionalmente hablando, por lo que se suele restringir su uso al camino más largo a partir del nodo (hasta la hoja) que estamos podando. Como se comentó anteriormente, el proceso de podado comienza en las hojas y continúa hacia la raíz pero, la cuestión es cómo decidir reemplazar un nodo interno por una hoja (*replacement*) o reemplazar un nodo interno por uno de sus nodos *hijo* (*raising*). Lo que se hace es comparar el error estimado de clasificación en el nodo en el que nos encontramos y compararlo con el error en cada uno de sus hijos y en su padre para realizar alguna de las operaciones o ninguna. En la figura 2.21 se muestra el pseudocódigo del proceso de podado que se emplea en C4.5.

```
Podar (raíz) {
  Si raíz No es HOJA Entonces
    Para cada hijo H de raíz Hacer
      Podar (H)

    Obtener Brazo más largo (B) de raíz // raising
    ErrorBrazo = EstimarErrorArbol (B, raíz.ejemplos)

    ErrorHoja = EstimarError (raíz, raíz.ejemplos) // replacement

    ErrorÁrbol = EstimarErrorArbol (raíz, raíz.ejemplos)

    Si ErrorHoja <= ErrorÁrbol Entonces // replacement
      raíz es Hoja
      Fin Poda

    Si ErrorBrazo <= ErrorÁrbol Entonces // raising
      raíz = B
      Podar (raíz)
}

EstimarErrorArbol (raíz, ejemplos) {
  Si raíz es HOJA Entonces
```

```
      EstimarError (raíz, ejemplos)
   Si no
      Distribuir los ejemplos (ej[]) en los brazos
      Para cada brazo (B)
         error = error + EstimarErrorArbol (B, ej[B])
}
```

Figura 3.19: *Pseudocódigo del algoritmo de podado en C4.5.*

De esta forma, el *subtree raising* se emplea únicamente para el subárbol más largo. Además, para estimar su error se emplean los ejemplos de entrenamiento, pero los del nodo origen, ya que si se *eleva* deberá clasificarlos él. En cuanto a la función *EstimarError*, es la función que estima el error de clasificación de una hoja del árbol. Así, para tomar la decisión debemos estimar el error de clasificación en un nodo determinado para un conjunto de test independiente. Habrá que estimarlo tanto para los nodos hoja como para los internos (suma de errores de clasificación de sus hijos). No se puede tomar como dato el error de clasificación en el conjunto de entrenamiento dado que, lógicamente, el error se subestimaría.

Una técnica para estimar el error de clasificación es la denominada *reduced-error pruning*, que consiste en dividir el conjunto de entrenamiento en *n* subconjuntos *n-1* de los cuáles servirán realmente para el entrenamiento del sistema y *1* para la estimación del error. Sin embargo, el problema es que la construcción del clasificador se lleva a cabo con menos ejemplos. Esta no es la técnica empleada en C4.5. La técnica empleada en C4.5 consiste en estimar el error de clasificación basándose en los propios ejemplos de entrenamiento. Para ello, en el nodo donde queramos estimar el error de clasificación, se toma la clase mayoritaria de sus ejemplos como clase representante. Esto implica que habrá *E* errores de clasificación de un total de *N* ejemplos que se clasifican a través de dicho nodo. El error observado será *f=E/N*, siendo *q* la probabilidad de error de clasificación del nodo y *p=1-q* la probabilidad de éxito. Se supone que la función *f* sigue una distribución binomial de parámetro *q*. Y lo que se desea obtener es el error *e*, que será la probabilidad del extremo superior con un intervalo *[f-z, f+z]* de confianza *c*. Dado que se trata de una distribución binomial, se obtendrá *e* mediante las ecuaciones 2.58 y 2.59.

$$P\left[\frac{f-q}{q(1-q)/N} \leq z\right] = c$$ Ec. 2.58

$$e = \left(\frac{f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}} \right)$$ Ec. 2.59

Como factor *c* (factor de confianza) se suele emplear en C4.5 el 25%, dado que es el que mejores resultados suele dar y que corresponde a un *z=0.69*.

Obtención de Reglas de Clasificación

Cualquier árbol de decisión se puede convertir en reglas de clasificación, entendiendo como tal una estructura del tipo *Si <Condición> Entonces <Clase>*. El algoritmo de generación de reglas consiste básicamente en, por cada rama del árbol de decisión, las preguntas y sus valores estarán en la parte izquierda de las reglas y la etiqueta del nodo hoja correspondiente en la parte derecha (clasificación). Sin embargo, este procedimiento generaría un sistema de reglas con mayor complejidad de la necesaria. Por ello, el sistema C4.5 [QUIN93] realiza un podado de las reglas obtenidas. En la figura 2.22 se muestra el algoritmo completo de obtención de reglas.

```
ObtenerReglas (árbol) {
  Convertir el árbol de decisión (árbol) a un conjunto de reglas, R
  error = error de clasificación con R
  Para cada regla Ri de R Hacer
    Para cada precondición pj de Ri Hacer
      nuevoError = error al eliminar pj de Ri
      Si nuevoError <= error Entonces
        Eliminar pj de Ri
        error = nuevoError
    Si Ri no tiene precondiciones Entonces
      Eliminar Ri
}
```

Figura 3.20: Pseudocódigo del algoritmo de obtención de reglas de C4.5.

En cuanto a la estimación del error, se realiza del mismo modo que para realizar el podado del árbol de decisión.

- ***Decision Stump*** **(Árbol de un solo nivel*)***

Todavía existe un algoritmo más sencillo que genera un árbol de decisión de un único nivel. Se trata de un algoritmo, [decision stump], que utiliza un único atributo para construir el árbol de decisión. La elección del único atributo que formará parte del árbol se realizará basándose en la ganancia de información, y a pesar de su simplicidad, en algunos problemas puede llegar a conseguir resultados interesantes. No tiene opciones de configuración, pero la implementación es muy completa, dado que admite tanto atributos numéricos como simbólicos y clases de ambos tipos también. El árbol de decisión tendrá tres ramas: una de ellas será para el caso de que el atributo sea desconocido, y las otras dos serán para el caso de que el valor del atributo del ejemplo de test sea igual a un valor concreto del atributo o distinto a dicho valor, en caso de los atributos simbólicos, o que el valor del ejemplo de test sea mayor o menor a un determinado valor en el caso de atributos numéricos. En el caso de los atributos simbólicos se considera cada valor posible del mismo y se calcula la ganancia de información con el atributo igual al valor, distinto al valor y valores desconocidos del atributo. En el caso de atributos simbólicos se busca el mejor punto de ruptura, tal y como se vio en el sistema C4.5. Deben tenerse en cuenta cuatro posibles casos al calcular la ganancia de información: que sea un atributo simbólico y la clase sea simbólica o que la clase sea numérica, o que sea un atributo numérico y la clase sea simbólica o que la clase sea numérica. A continuación se comenta cada caso por separado.

Atributo Simbólico y Clase Simbólica

Se toma cada vez un valor v_x del atributo simbólico A_i como base y se consideran únicamente tres posibles ramas en la construcción del árbol: que el atributo A_i sea igual a v_x, que el atributo A_i sea distinto a v_x o que el valor del atributo A_i sea desconocido. Con ello, se calcula la entropía del atributo tomando como base el valor escogido tal y como se muestra en la ecuación 2.60, en la que el valor de *j* en el sumatorio va desde *1* a *3* porque los valores del atributo se restringen a tres: igual a v_x , distinto de v_x o valor desconocido. En cuanto a los parámetros, n_{ij} es el número de ejemplos con valor *j* en el atributo *i*, n el número total de ejemplos y n_{ijk} el número de ejemplos con valor *j* en el atributo *i* y que pertenece a la clase *k*.

$$I(A_{iv_x}) = \frac{\sum_{j=1}^{3} n_{ij} log\left(n_{ij}\right) - I_{ij}}{n}; \; I_{ij} = -\sum_{k=1}^{nc} n_{ijk} log\left(n_{ijk}\right) \qquad \text{Ec. 2.60}$$

Atributo Numérico y Clase Simbólica

Se ordenan los ejemplos según el atributo A_i y se considera cada valor v_x del atributo como posible punto de corte. Se consideran entonces como posibles valores del atributo el rango menor o igual a v_x, mayor a v_x y valor desconocido. Se calcula la entropía del rango tomando como base esos tres posibles valores restringidos del atributo.

Atributo Simbólico y Clase Numérica

Se vuelve a tomar como base cada vez cada valor del atributo, tal y como se hacía en el caso *Atributo Simbólico y Clase Simbólica*, pero en este caso se calcula la varianza de la clase para los valores del atributo mediante la ecuación 2.61, donde S_j es la suma de los valores de la clase de los ejemplos con valor j en el atributo i, SS_j es la suma de los valores de la clase al cuadrado y W_j es la suma de los pesos de los ejemplos (número de ejemplos si no se incluyen pesos) con valor j en el atributo.

$$Varianza(A_{iv_x}) = \sum_{j=1}^{3} \left(SS_j - \frac{S_j}{W_j} \right) \qquad \text{Ec. 2.61}$$

Atributo Numérico y Clase Numérica

Se considera cada valor del atributo como punto de corte tal y como se hacía en el caso *Atributo Numérico y Clase Simbólica*. Posteriormente, se calcula la varianza tal y como se muestra en la ecuación 2.61.

En cualquiera de los cuatro casos que se han comentado, lo que se busca es el valor mínimo de la ecuación calculada, ya sea la entropía o la varianza. De esta forma se obtiene el atributo que será raíz del árbol de decisión y sus tres ramas. Lo único que se hará por último es construir dicho árbol: cada rama finaliza en un nodo *hoja* con el valor de la clase, que será la media o la moda de los ejemplos que se clasifican por ese camino, según se trate de una clase numérica o simbólica.

3.5.3. Reglas de Clasificación

Las técnicas de Inducción de Reglas [QUIN87, QUIN93] surgieron hace más de dos décadas y permiten la generación y contraste de árboles de decisión, o reglas y patrones a partir de los datos de entrada. La información de entrada será un conjunto de casos donde se ha asociado una clasificación o evaluación a un conjunto de variables o atributos. Con esa información estas técnicas obtienen el árbol de decisión o conjunto de reglas que soportan la evaluación o clasificación [CN89, HMM86]. En los casos en que la información de entrada posee algún tipo de “ruido" o defecto (insuficientes atributos o datos, atributos irrelevantes o errores u omisiones en los datos) estas técnicas pueden habilitar métodos estadísticos de tipo probabilístico para generar árboles de decisión recortados o podados. También en estos casos pueden identificar los atributos irrelevantes, la falta de atributos discriminantes o detectar "gaps" o huecos de conocimiento. Esta técnica suele llevar asociada una alta interacción con el analista de forma que éste pueda intervenir en cada paso de la construcción de las reglas, bien para aceptarlas, bien para modificarlas [MM95].

La inducción de reglas se puede lograr fundamentalmente mediante dos caminos: Generando un árbol de decisión y extrayendo de él las reglas [QUIN93], como puede hacer el sistema C4.5 o bien mediante una estrategia de *covering*, consistente en tener en cuenta cada vez una clase y buscar las reglas necesarias para cubrir [cover] todos los ejemplos de esa clase; cuando se obtiene una regla se eliminan todos los ejemplos que cubre y se continúa buscando más reglas hasta que no haya más ejemplos de la clase. A continuación se muestran una técnica de inducción de reglas basada en árboles de decisión, otra basada en *covering* y una más que mezcla las dos estrategias.

- **Algoritmo 1R**

El más simple algoritmo de reglas de clasificación para un conjunto de ejemplos es el 1R [HOL93]. Este algoritmo genera un árbol de decisión de un nivel expresado mediante reglas. Consiste en seleccionar un atributo (nodo raíz) del cual nace una rama por cada valor, que va a parar a un nodo hoja con la clase más probable de los ejemplos de entrenamiento que se clasifican a través suyo. Este algoritmo se muestra en la figura 2.23.

```
1R (ejemplos) {
  Para cada atributo (A)
    Para cada valor del atributo (Ai)
      Contar el número de apariciones de cada clase con Ai
      Obtener la clase más frecuente (Cj)
      Crear una regla del tipo Ai -> Cj
    Calcular el error de las reglas del atributo A
  Escoger las reglas con menor error
}
```

Figura 3.21: Pseudocódigo del algoritmo 1R.

La clase debe ser simbólica, mientras los atributos pueden ser simbólicos o numéricos. También admite valores desconocidos, que se toman como otro valor más del atributo. En cuanto al error de las reglas de un atributo, consiste en la proporción entre los ejemplos que cumplen la regla y los ejemplos que cumplen la premisa de la regla. En el caso de los atributos numéricos, se generan una serie de *puntos de ruptura* [breakpoint], que discretizarán dicho atributo formando conjuntos. Para ello, se ordenan los ejemplos por el atributo numérico y se recorren. Se van contando las apariciones de cada clase hasta un número *m* que indica el mínimo número de ejemplos que pueden pertenecer a un conjunto, para evitar conjuntos demasiado pequeños. Por último, se unen a este conjunto ejemplos con la clase más frecuente y ejemplos con el mismo valor en el atributo.

La sencillez de este algoritmo es un poco insultante. Su autor llega a decir [HOL93; pag 64] : *“Program 1R is ordinary in most respects.”* Tanto es así que 1R no tiene ningún elemento de sofistificación y genera para cada atributo un árbol de profundidad 1, donde una rama está etiquetada por *missing* si es que aparecen valores desconocidos (*missing values)* en ese atributo en el conjunto de entrenamiento; el resto de las ramas tienen como etiqueta un intervalo construido de una manera muy simple, como se ha explicado antes, o un valor nominal, según el tipo de atributo del que se trate. Lo sorprendente de este sistema es su rendimiento. En [HOL93] se describen rendimientos que en media están por debajo de los de C4.5 en 5,7 puntos porcentuales de aciertos de clasificación. Para la realización de las pruebas, Holte, elige un conjunto de 16 problemas del almacén de la U.C.I. [Blake, Keog, Merz, 98] que desde entonces han gozado de cierto reconocimiento como conjunto de pruebas; en alguno de estos problemas introduce algunas modificaciones que también se han hecho estándar. El mecanismo de estimación consiste en separar el subconjunto de entrenamiento original en subconjuntos de entrenamiento y test en

proporción 2/3 y 1/3 respectivamente y repetir el experimento 25 veces. Aunque la diferencia de 5,7 es algo elevada, en realidad en 14 de los 16 problemas la diferencia es solo de 3,1 puntos. En la tabla 2.5 se presenta un ejemplo de 1R, basado en los ejemplos de la tabla 2.1.

Tabla2.5. Resultados del algoritmo 1R.

atributo	***reglas***	***errores***	***error total***
vista	Soleado → no Nublado → si Lluvioso → si	2/5 0/4 2/5	4/14
temperatura	Alta → no Media → si Baja → si	2/4 2/6 1/4	5/14
humedad	Alta → no Normal → si	3/7 1/7	4/14
viento	Falso → si Cierto → no	2/8 3/6	5/14

Para clasificar según la clase jugar, 1R considera cuatro conjuntos de reglas, uno por cada atributo, que son las mostradas en la tabla anterior, en las que además aparecen los errores que se cometen. De esta forma se concluye que como los errores mínimos corresponden a las reglas generadas por los atributos vista y humedad, cualquiera de ellas es valida, de manera que arbitrariamente se puede elegir cualquiera de estos dos conjuntos de reglas como generador de *1R*.

- **Algoritmo PRISM**

PRISM [CEN87] es un algoritmo básico de aprendizaje de reglas que asume que no hay ruido en los datos. Sea *t* el número de ejemplos cubiertos por la regla y *p* el número de ejemplos positivos cubiertos por la regla. Lo que hace PRISM es añadir condiciones a reglas que maximicen la relación *p/t* (relación entre los ejemplos positivos cubiertos y ejemplos cubiertos en total). En la figura 2.24 se muestra el algoritmo de PRISM.

```
PRISM (ejemplos) {
  Para cada clase (C)
    E = ejemplos
    Mientras E tenga ejemplos de C
      Crea una regla R con parte izquierda vacía y clase C
      Hasta R perfecta Hacer
        Para cada atributo A no incluido en R y cada valor v de A
          Considera añadir la condición A=v a la parte izquierda de R
          Selecciona el par A=v que maximice p/t
            (en caso de empates, escoge la que tenga p mayor)
        Añadir A=v a R
    Elimina de E los ejemplos cubiertos por R
```

Figura 3.22: Pseudocódigo del algoritmo PRISM.

Este algoritmo va eliminando los ejemplos que va cubriendo cada regla, por lo que las reglas tienen que interpretarse en orden. Se habla entonces de listas de reglas [decision list]. En la figura 2.25 se muestra un ejemplo de cómo actúa el algoritmo. Concretamente se trata de la aplicación del mismo sobre el ejemplo de la tabla 2.1.

Regla 1. Clase "Sí".

Añadir a "If <vacío> Then Sí"	p/t
Vista = Soleado	2/5
Vista = Nublado	4/4
Vista = Lluvioso	3/5
Temperatura = Alta	2/4
Temperatura = Media	4/6
Temperatura = Baja	3/4
Humedad = Alta	3/7
Humedad = Normal	6/7
Viento = Sí	3/6
Viento = No	6/8

If Vista = Nublado **Then** Sí

Regla 2. Clase "Sí".

Añadir a "If <vacío> Then Sí"	p/t
Vista = Soleado	2/5
Vista = Lluvioso	3/5
Temperatura = Alta	0/2
Temperatura = Media	3/5
Temperatura = Baja	2/3
Humedad = Alta	1/5
Humedad = Normal	4/5
Viento = Sí	1/4
Viento = No	4/6

Añadir a "If Humedad=Normal Then Sí"	p/t
Vista = Soleado	2/2
Vista = Lluvioso	2/3
Temperatura = Alta	0/0
Temperatura = Media	2/2
Temperatura = Baja	2/3
Viento = Sí	1/2
Viento = No	3/3

If Humedad=Normal **and** Viento = No **Then** Sí

Lista de Decisión Completa:

If Vista = Nublado **Then** Sí
If Humedad=Normal **and** Viento = No **Then** Sí
If Temperatura = Media **and** Humedad = Normal **Then** Sí
If Vista = Lluvioso **and** Viento = No **Then** Sí
If Vista = Soleado **and** Humedad = Alta **Then** No
If Vista = Lluvioso **and** Viento = Sí **Then** Sí

Figura 3.23: Ejemplo de PRISM.

En la figura 2.25 se muestra cómo el algoritmo toma en primer lugar la clase *Sí*. Partiendo de todos los ejemplos de entrenamiento (un total de catorce) calcula el cociente *p/t* para cada par atributo-valor y escoge el mayor. En este caso, dado que la condición escogida hace la regla perfecta (*p/t = 1*), se eliminan los cuatro ejemplos que cubre dicha regla y se busca una nueva regla. En la segunda regla se obtiene en un primer momento una condición que no hace perfecta la regla, por lo que se continúa buscando con otra condición. Finalmente, se muestra la lista de decisión completa que genera el algoritmo.

- **Algoritmo PART**

Uno de los sistemas más importantes de aprendizaje de reglas es el proporcionado por C4.5 [QUI93], explicado anteriormente. Este sistema, al igual que otros sistemas de inducción de reglas, realiza dos fases: primero, genera un conjunto de reglas de clasificación y después refina estas reglas para mejorarlas, realizando así una proceso de optimización global de dichas reglas. Este proceso de optimización global es siempre muy complejo y costoso computacionalmente hablando. Por otro lado, el algoritmo PART [FRWI98] es un sistema que obtiene reglas sin dicha optimización global. Recibe el nombre PART por su modo de actuación: *obtaining rules from PARTial decision trees*, y fue desarrollado por el grupo neozelandés que construyó el entorno WEKA [WF98].

El sistema se basa en las dos estrategias básicas para la inducción de reglas: el *covering* y la generación de reglas a partir de árboles de decisión. Adopta la estrategia del *covering* (con lo que se obtiene una lista de decisión) dado que genera una regla, elimina los ejemplares que dicha regla cubre y continúa generando reglas hasta que no queden ejemplos por clasificar. Sin embargo, el proceso de generación de cada regla no es el usual. En este caso, para crear una regla, se genera un árbol de decisión podado, se obtiene la *hoja* que clasifique el mayor número de ejemplos, que se transforma en la regla, y posteriormente se elimina el árbol. Uniendo estas dos estrategias se consigue mayor flexibilidad y velocidad. Además, no se genera un árbol completo, sino un árbol parcial [partial decisión tree]. Un árbol parcial es un árbol de decisión que contiene *brazos* con subárboles no definidos. Para generar este árbol se integran los procesos de construcción y podado hasta que se encuentra un subárbol *estable* que no puede simplificarse más, en cuyo caso se para el proceso y se genera la regla a partir de dicho subárbol. Este proceso se muestra en la figura 2.26.

```
Expandir (ejemplos) {
  elegir el mejor atributo para dividir en subconjuntos
  Mientras (subconjuntos No expandidos)
             Y (todos los subconjuntos expandidos son HOJA)
    Expandir (subconjunto)
  Si (todos los subconjuntos expandidos son HOJA)
             Y (errorSubárbol >= errorNodo)
    deshacer la expansión del nodo y nodo es HOJA
```

Figura 3.24: Pseudocódigo de expansión de PART.

El proceso de elección del mejor atributo se hace como en el sistema C4.5, esto es, basándose en la razón de ganancia. La expansión de los subconjuntos generados se realiza en orden, comenzando por el que tiene menor entropía y finalizando por el que tiene mayor. La razón de realizarlo así es porque si un subconjunto tiene menor entropía hay más probabilidades de que se genere un subárbol menor y consecuentemente se cree una regla más general. El proceso continúa recursivamente expandiendo los subconjuntos hasta que se obtienen *hojas*, momento en el que se realizará una vuelta atrás [backtracking]. Cuando se realiza dicha vuelta atrás y los hijos del nodo en cuestión son *hojas*, comienza el podado tal y como se realiza en C4.5 (comparando el error esperado del subárbol con el del nodo), pero únicamente se realiza la función de reemplazamiento del nodo por hoja [subtree replacement]. Si se realiza el podado se realiza otra vuelta atrás hacia el nodo *padre*, que sigue explorando el resto de sus *hijos*, pero si no se puede realizar el podado el *padre* no continuará con la exploración del resto de nodos hijos (ver segunda condición del bucle "*mientras"* en la figura 2.26). En este momento finalizará el proceso de expansión y generación del árbol de decisión.

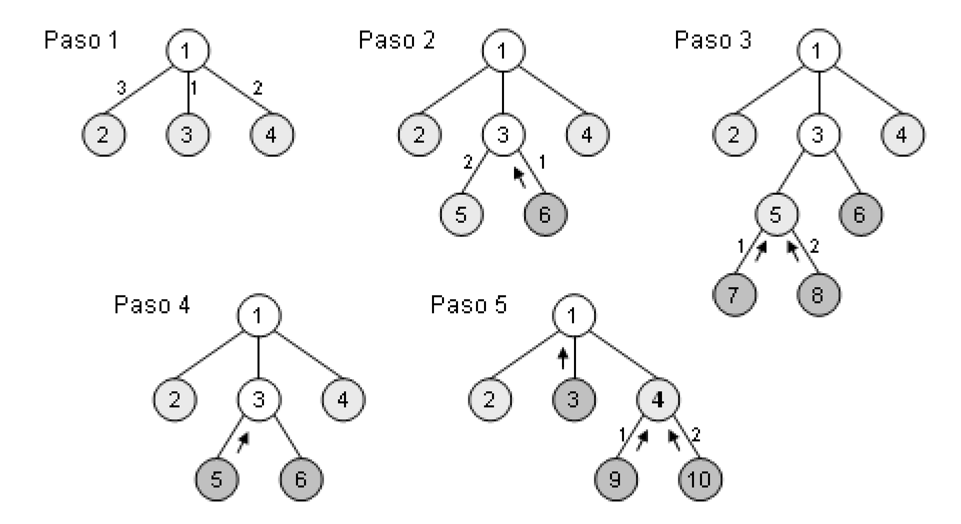


Figura 3.25: Ejemplo de generación de árbol parcial con PART.

En la figura 2.27 se presenta un ejemplo de generación de un árbol parcial donde, junto a cada brazo de un nodo, se muestra el orden de exploración (orden ascendente según el valor de la entropía). Los nodos con relleno gris claro son los que aún no se han explorado y los nodos con relleno gris oscuro los nodos *hoja*. Las flechas ascendentes representan el proceso de *backtracking*. Por último, en el paso 5, cuando el nodo 4 es explorado y los nodos 9 y 10 pasan a ser hoja, el nodo *padre* intenta realizar el proceso de podado, pero no se realiza el reemplazo (representado con el 4 en negrita), con lo que el proceso, al volver al nodo 1, finaliza sin explorar el nodo 2.

Una vez generado el árbol parcial se extrae una regla del mismo. Cada *hoja* se corresponde con una posible regla, y lo que se busca es la mejor *hoja*. Si bien se pueden considerar otras heurísticas, en el algoritmo PART se considera mejor hoja aquella que cubre un mayor número de ejemplos. Se podría haber optado, por ejemplo, por considerar mejor aquella que tiene un menor error esperado, pero tener una regla muy precisa no significa lograr un conjunto de reglas muy preciso. Por último, PART permite que haya atributos con valores desconocidos tanto en el proceso de aprendizaje como en el de validación y atributos numéricos, tratándolos exactamente como el sistema C4.5.

3.5.4. Clasificación Bayesiana

Los clasificadores Bayesianos [DH73] son clasificadores estadísticos, que pueden predecir tanto las probabilidades del número de miembros de clase, como la probabilidad de que una muestra dada pertenezca a una clase particular. La clasificación Bayesiana se basa en el teorema de Bayes, y los clasificadores Bayesianos han demostrado una alta exactitud y velocidad cuando se han aplicado a grandes bases de datos Diferentes estudios comparando los algoritmos de clasificación han determinado que un clasificador Bayesiano sencillo conocido como el clasificador “*naive* Bayesiano” [JOH97] es comparable en rendimiento a un árbol de decisión y a clasificadores de redes de neuronas. A continuación se explica los fundamentos de los clasificadores bayesianos y, más concretamente, del clasificador *naive* Bayesiano. Tras esta explicación se comentará otro clasificador que, si bien no es un clasificador bayesiano, esta relacionado con él, dado que se trata también de un clasificador basado en la estadística.

- **Clasificador *Naive Bayesiano***

Lo que normalmente se quiere saber en aprendizaje es cuál es la mejor hipótesis (más probable) dados los datos. Si denotamos *P(D)* como la probabilidad a priori de los datos (i.e., cuales datos son más probables que otros), *P(D|h)* la probabilidad de los datos dada una hipótesis, lo que queremos estimar es: *P(h|D)*, la probabilidad posterior de *h* dados los datos. Esto se puede estimar con el teorema de Bayes, ecuación 2.62.

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)} \qquad \text{Ec. 2.62}$$

Para estimar la hipótesis más probable (MAP, [maximum a posteriori hipótesis]) se busca el mayor *P(h|D)* como se muestra en la ecuación 2.63.

$$\begin{aligned} h_{MAP} &= argmax_{h \in H}\left(P(h \mid D)\right) \\ &= argmax_{h \in H}\left(\frac{P(D \mid h)P(h)}{P(D)}\right) \\ &= argmax_{h \in H}\left(P(D \mid h)P(h)\right) \end{aligned} \qquad \text{Ec. 2.63}$$

Ya que *P(D)* es una constante independiente de *h*. Si se asume que todas las hipótesis son igualmente probables, entonces resulta la hipótesis de máxima verosimilitud (ML, [maximum likelihood]) de la ecuación 2.64.

$$h_{ML} = argmax_{h \in H}\left(P(D \mid h)\right) \qquad \text{Ec. 2.64}$$

El clasificador *naive* [ingenuo] Bayesiano se utiliza cuando se quiere clasificar un ejemplo descrito por un conjunto de atributos (a_i's) en un conjunto finito de clases (*V*). Clasificar un nuevo ejemplo de acuerdo con el valor más probable dados los valores de sus atributos. Si se aplica 2.64 al problema de la clasificación se obtendrá la ecuación 2.65.

$$\begin{aligned} v_{MAP} &= argmax_{v_j \in V}\left(P(v_j \mid a_1,...,a_n)\right) \\ &= argmax_{v_j \in V}\left(\frac{P(a_1,...,a_n \mid v_j)P(v_j)}{P(a_1,...,a_n)}\right) \\ &= argmax_{v_j \in V}\left(P(a_1,...,a_n \mid v_j)P(v_j)\right) \end{aligned} \qquad \text{Ec. 2.65}$$

Además, el clasificador *naive* Bayesiano asume que los valores de los atributos son condicionalmente independientes dado el valor de la clase, por lo que se hace cierta la ecuación 2.66 y con ella la 2.67.

$$P(a_1,...,a_n \mid v_j) = \prod_i P(a_i \mid v_j) \qquad \text{Ec. 2.66}$$

$$P(v_j \mid a_1,\ldots,a_n) = P(v_j) \times \prod_i P(a_i \mid v_j) \qquad \text{Ec. 2.67}$$

Los clasificadores *naive* Bayesianos asumen que el efecto de un valor del atributo en una clase dada es independiente de los valores de los otros atributos. Esta suposición se llama “independencia condicional de clase”. Ésta simplifica los cálculos involucrados y, en este sentido, es considerado "ingenuo” [naive]. Esta asunción es una simplificación de la realidad. A pesar del nombre del clasificador y de la simplificación realizada, el *naive* Bayesiano funciona muy bien, sobre todo cuando se filtra el conjunto de atributos seleccionado para eliminar redundancia, con lo que se elimina también dependencia entre datos. En la figura 2.28 se muestra un ejemplo de aprendizaje con el clasificador *naive Bayesiano*, así como una muestra de cómo se clasificaría un ejemplo de test. Como ejemplo se empleará el de la tabla 2.1.

Proceso de Aprendizaje

Vista			Temperatura			Humedad			Viento			Jugar	
	Si	No		Si	No		Si	No		Si	No	Si	No
Soleado	2	3	Alta	2	2	Alta	3	4	Si	3	3	9	5
Nublado	4	0	Media	4	2	Normal	6	1	No	6	2		
Lluvioso	3	2	Baja	3	1								
Soleado	2/9	3/5	Alta	2/9	2/5	Alta	3/9	4/5	Si	3/9	3/5	9/14	5/14
Nublado	4/9	0/5	Media	4/9	2/5	Normal	6/9	1/5	No	6/9	2/5		
Lluvioso	3/9	2/5	Baja	3/9	1/5								

Clasificación de un ejemplo de test

Vista	Temperatura	Humedad	Viento	Jugar
Soleado	Fría	Alta	Si	¿

$$P(Si \mid E) = P(Si) \times \prod_i P(a_i \mid Si) = \frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} = 0{,}0053$$

$$P(No \mid E) = P(No) \times \prod_i P(a_i \mid No) = \frac{5}{14} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} = 0{,}0206$$

Normalizado

$$P(Si \mid E) = \frac{0{,}0053}{0{,}0053+0{,}0206} = 20{,}5\%$$

$$P(No \mid E) = \frac{0{,}0206}{0{,}0053+0{,}0206} = 79{,}5\%$$

Figura 3.26: Ejemplo de aprendizaje y clasificación con naive Bayesiano.

En este ejemplo se observa que en la fase de aprendizaje se obtienen todas las probabilidades condicionadas $P(a_i|v_j)$ y las probabilidades $P(v_j)$. En la clasificación se realiza el productorio y se escoge como clase del ejemplo de entrenamiento la que obtenga un mayor valor. Algo que puede ocurrir durante el entrenamiento con este clasificador es que para cada valor de cada atributo no se encuentren ejemplos para todas las clases. Supóngase que para el atributo a_i y el valor j de dicho atributo no hay ningún ejemplo de entrenamiento con clase k. En este caso, $P(a_{ij}|k)=0$. Esto hace que si se intenta clasificar cualquier ejemplo con el par atributo-valor a_{ij}, la probabilidad asociada para la clase k será siempre 0, ya que hay que realizar el productorio de las probabilidades condicionadas para todos los atributos de la instancia. Para resolver este problema se parte de que las probabilidades se contabilizan a partir de las frecuencias de aparición de cada evento o, en nuestro caso, las frecuencias de aparición de cada terna atributo-valor-clase. El *estimador de Laplace*, consiste en

comenzar a contabilizar la frecuencia de aparición de cada terna a partir del *1* y no del *0*, con lo que ninguna probabilidad condicionada será igual a *0*.

Una ventaja de este clasificador es la cuestión de los valores perdidos o desconocidos: en el clasificador *naive* Bayesiano si se intenta clasificar un ejemplo con un atributo sin valor simplemente el atributo en cuestión no entra en el productorio que sirve para calcular las probabilidades. Respecto a los atributos numéricos, se suele suponer que siguen una distribución Normal o Gaussiana. Para estos atributos se calcula la media μ y la desviación típica σ obteniendo los dos parámetros de la distribución $N(\mu, \sigma)$, que sigue la expresión de la ecuación 2.68, donde el parámetro x será el valor del atributo numérico en el ejemplo que se quiere clasificar.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad \text{Ec. 2.68}$$

- **Votación por intervalos de características**

Este algoritmo es una técnica basada en la proyección de características. Se le denomina “votación por intervalos de características” (VFI, [Voting Feature Interval]) porque se construyen intervalos para cada característica [feature] o atributo en la fase de aprendizaje y el intervalo correspondiente en cada característica “vota” para cada clase en la fase de clasificación. Al igual que en el clasificador *naive* Bayesiano, cada característica es tratada de forma individual e independiente del resto. Se diseña un sistema de votación para combinar las clasificaciones individuales de cada atributo por separado.

Mientras que en el clasificador *naive* Bayesiano cada característica participa en la clasificación asignando una probabilidad para cada clase y la probabilidad final para cada clase consiste en el producto de cada probabilidad dada por cada característica, en el algoritmo VFI cada característica distribuye sus votos para cada clase y el voto final de cada clase es la suma de los votos obtenidos por cada característica. Una ventaja de estos clasificadores, al igual que ocurría con el clasificador *naive* Bayesiano, es el tratamiento de los valores desconocidos tanto en el proceso de aprendizaje como en el de clasificación: simplemente se ignoran, dado que se considera cada atributo como independiente del resto.

En la fase de aprendizaje del algoritmo VFI se construyen intervalos para cada atributo contabilizando, para cada clase, el número de ejemplos de entrenamiento que aparecen en dicho intervalo. En la fase de clasificación, cada atributo del ejemplo de test añade votos para cada clase dependiendo del intervalo en el que se encuentre y el conteo de la fase de aprendizaje para dicho intervalo en cada clase. En la figura 2.29 se muestra este algoritmo.

```
Aprendizaje (ejemplos) {
  Para cada atributo (A) Hacer
```

```
    Si A es NUMÉRICO Entonces
      Obtener mínimo y máximo de A para cada clase en ejemplos
      Ordenar los valores obtenidos (I intervalos)
    Si no /* es SIMBÓLICO */
      Obtener los valores que recibe A para cada clase en ejemplos
      Los valores obtenidos son puntos (I intervalos)

    Para cada intervalo I Hacer
      Para cada clase C Hacer
        contadores [A, I, C] = 0

    Para cada ejemplo E Hacer
      Si A es conocido Entonces
        Si A es SIMBÓLICO Entonces
          contadores [A, E.A, E.C] += 1
        Si no /* es NUMÉRICO */
          Obtener intervalo I de E.A
          Si E.A = extremo inferior de intervalo I Entonces
            contadores [A, I, E.C] += 0.5
            contadores [A, I-1, E.C] += 0.5
          Si no
            contadores [A, I, E.C] += 1

    Normalizar contadores[] /* Σc contadores[A, I, C] = 1 */
}

clasificar (ejemplo E) {
  Para cada atributo (A) Hacer
    Si E.A es conocido Entonces
      Si A es SIMBÓLICO
        Para cada clase C Hacer
          voto[A, C] = contadores[A, E.A, C]
      Si no /* es NUMÉRICO */
        Obtener intervalo I de E.A
        Si E.A = límite inferior de I Entonces
          Para cada clase C Hacer
            voto[A, C] = 0.5*contadores[A,I,C] +
                        0.5*contadores[A,I-1,C]
        Si no
          Para cada clase C Hacer
            voto[A, C] = contadores [A, I, C]

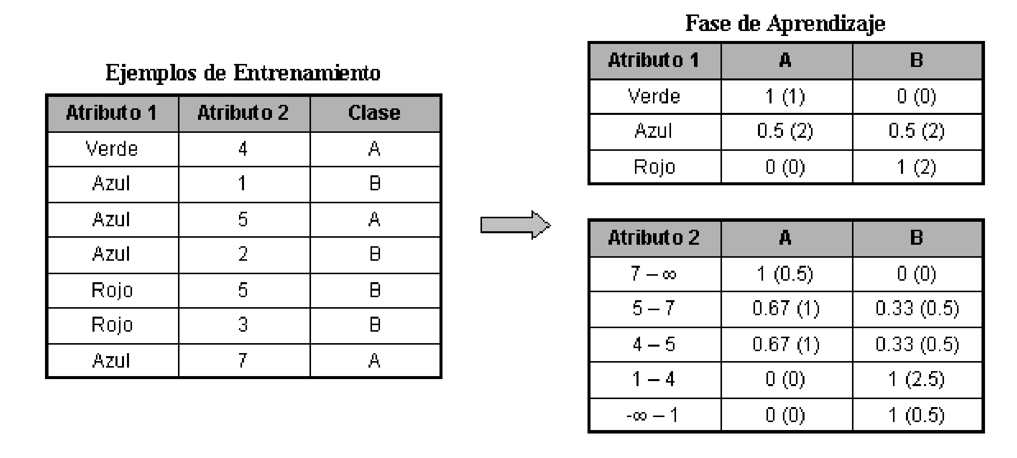
    voto[C] = voto[C] +  voto[A, C]

  Normalizar voto[]/* Σc voto[C] = 1 */
```

Figura 3.27: Pseudocódigo del algoritmo VFI.

En la figura 2.30 se presenta un ejemplo de entrenamiento y clasificación con el algoritmo VFI, en el que se muestra una tabla con los ejemplos de entrenamiento y cómo el proceso de aprendizaje consiste en el establecimiento de intervalos para cada atributo con el conteo de ejemplos que se encuentran en cada intervalo. Se muestra entre paréntesis el número de ejemplos que se encuentran en la clase e intervalo concreto, mientras que fuera de los paréntesis se encuentra el valor normalizado. Para el atributo simbólico simplemente se toma como intervalo (punto) cada valor de dicho atributo y se cuenta el número de ejemplos que tienen un valor determinado en el atributo para la clase del ejemplo en cuestión. En el caso del atributo numérico, se obtiene el máximo y el mínimo valor del atributo para cada clase que en este caso son *4* y *7* para la clase *A*, y *1* y *5* para la clase *B*. Se ordenan los valores formándose un

total de cinco intervalos y se cuenta el número de ejemplos que se encuentran en un intervalo determinado para su clase, teniendo en cuenta que si se encuentra en el punto compartido por dos intervalos se contabiliza la mitad para cada uno de ellos. También se muestra un ejemplo de clasificación: en primer lugar, se obtienen los votos que cada atributo por separado concede a cada clase, que será el valor normalizado del intervalo (o punto si se trata de atributos simbólicos) en el que se encuentre el valor del atributo, y posteriormente se suman los votos (que se muestra entre paréntesis) y se normaliza. La clase con mayor porcentaje de votos (en el ejemplo la clase *A*) *gana*.



Ejemplos de Entrenamiento

Atributo 1	Atributo 2	Clase
Verde	4	A
Azul	1	B
Azul	5	A
Azul	2	B
Rojo	5	B
Rojo	3	B
Azul	7	A

Fase de Aprendizaje

Atributo 1	A	B
Verde	1 (1)	0 (0)
Azul	0.5 (2)	0.5 (2)
Rojo	0 (0)	1 (2)

Atributo 2	A	B
7 – ∞	1 (0.5)	0 (0)
5 – 7	0.67 (1)	0.33 (0.5)
4 – 5	0.67 (1)	0.33 (0.5)
1 – 4	0 (0)	1 (2.5)
-∞ – 1	0 (0)	1 (0.5)

Clasificación de un Ejemplo de Test

Atributo 1	Atributo 2	Clase
Azul	6	?

Clase	Voto 1	Voto 2	Votos
A	0.5	0.67	**0.585 (1.17)**
B	0.5	0.33	0.415 (0.83)

Figura 3.28: Ejemplo de aprendizaje y clasificación con VFI.

3.5.5. Aprendizaje Basado en Ejemplares

El aprendizaje basado en ejemplares o instancias [BRIS96] tiene como principio de funcionamiento, en sus múltiples variantes, el almacenamiento de ejemplos: en unos casos todos los ejemplos de entrenamiento, en otros solo los más representativos, en otros los incorrectamente clasificados cuando se clasifican por primera vez, etc. La clasificación posterior se realiza por medio de una función que mide la proximidad o parecido. Dado un ejemplo para clasificar se le clasifica de acuerdo al ejemplo o ejemplos más próximos. El *bias* (sesgo) que rige este método es la proximidad; es decir, la generalización se guía por la proximidad de un ejemplo a otros. Algunos autores consideran este *bias* más apropiado para el aprendizaje de conceptos naturales que el correspondiente al proceso inductivo (Bareiss et al. en [KODR90]), por otra parte también se ha estudiado la relación entre este método y los que generan reglas (Clark, 1990).

Se han enumerado ventajas e inconvenientes del aprendizaje basado en ejemplares [BRIS96], pero se suele considerar no adecuado para el tratamiento de

atributos no numéricos y valores desconocidos. Las mismas medidas de proximidad sobre atributos simbólicos suelen proporcionar resultados muy dispares en problemas diferentes. A continuación se muestran dos técnicas de aprendizaje basado en ejemplares: el método de los *k*-vecinos más próximos y el *k* estrella.

- **Algoritmo de los k-vecinos más próximos**

El método de los *k*-vecinos más próximos [MITC97] (KNN, [k-Nearest Neighbor]) está considerado como un buen representante de este tipo de aprendizaje, y es de gran sencillez conceptual. Se suele denominar método porque es el esqueleto de un algoritmo que admite el intercambio de la función de proximidad dando lugar a múltiples variantes. La función de proximidad puede decidir la clasificación de un nuevo ejemplo atendiendo a la clasificación del ejemplo o de la mayoría de los *k* ejemplos más cercanos. Admite también funciones de proximidad que consideren el peso o coste de los atributos que intervienen, lo que permite, entre otras cosas, eliminar los atributos irrelevantes. Una función de proximidad clásica entre dos instancias x_i y x_j , si suponemos que un ejemplo viene representado por una *n*-tupla de la forma $(a_1(x), a_2(x), ... , a_n(x))$ en la que $a_r(x)$ es el valor de la instancia para el atributo a_r, es la distancia euclídea, que se muestra en la ecuación 2.69.

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^{n} \left(x_{il} - x_{jl}\right)^2} \qquad \text{Ec. 2.69}$$

En la figura 2.31 se muestra un ejemplo del algoritmo KNN para un sistema de dos atributos, representándose por ello en un plano. En este ejemplo se ve cómo el proceso de aprendizaje consiste en el almacenamiento de todos los ejemplos de entrenamiento. Se han representado los ejemplos de acuerdo a los valores de sus dos atributos y la clase a la que pertenecen (las clases son + y -). La clasificación consiste en la búsqueda de los *k* ejemplos (en este caso 3) más cercanos al ejemplo a clasificar. Concretamente, el ejemplo *a* se clasificaría como -, y el ejemplo *b* como +.

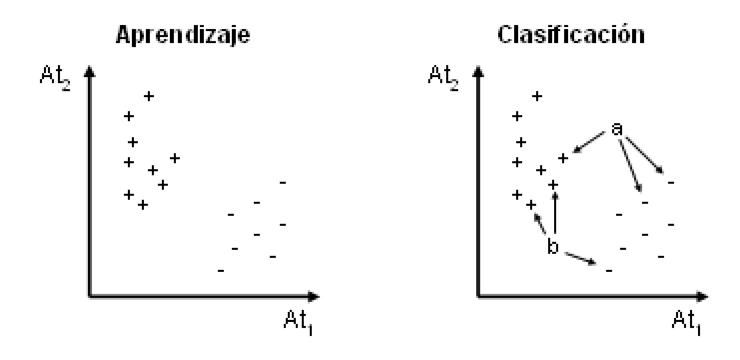


Figura 3.29: Ejemplo de Aprendizaje y Clasificación con KNN.

Dado que el algoritmo k-NN permite que los atributos de los ejemplares sean simbólicos y numéricos, así como que haya atributos sin valor [missing values] el

algoritmo para el cálculo de la distancia entre ejemplares se complica ligeramente. En la figura 2.32 se muestra el algoritmo que calcula la distancia entre dos ejemplares cualesquiera.

```
Distancia (E1, E2) {
   dst = 0
   n = 0
   Para cada atributo A Hacer {
      dif = Diferencia(E1.A, E2.A)
      dst = dst + dif * dif
      n = n + 1
   }
   dst = dst / n
   Devolver dst
}

Diferencia (A1, A2) {
   Si A1.nominal Entonces {
      Si SinValor(A1) O SinValor(A2) O A1 <> A2 Entonces
         Devolver 1
      Si no
         Devolver 0
   } Si no {
      Si SinValor(A1) O SinValor(A2) Entonces {
         Si SinValor(A1) Y SinValor(A2) Entonces
            Devolver 1
         Si SinValor(A1) Entonces
            dif = A2
         Si no Entonces
            dif = A1
         Si dif < 0.5 Entonces
            Devolver 1 – dif
         Si no
            Devolver dif
      } Si no
         Devolver abs(A1 – A2)
   }
}
```

Figura 3.30: Pseudocódigo del algoritmo empleado para definir la distancia entre dos ejemplos.

Además de los distintos tipos de atributos hay que tener en cuenta también, en el caso de los atributos numéricos, los rangos en los que se mueven sus valores. Para evitar que atributos con valores muy altos tengan mucho mayor peso que atributos con valores bajos, se normalizarán dichos valores con la ecuación 2.70.

$$\frac{x_{il} - min_l}{Max_l - min_l} \qquad \text{Ec. 2.70}$$

En esta ecuación x_{if} será el valor *i* del atributo *f*, siendo min_f el mínimo valor del atributo *f* y Max_f el máximo. Por otro lado, el algoritmo permite dar mayor preferencia a aquellos ejemplares más cercanos al que deseamos clasificar. En ese caso, en lugar de emplear directamente la distancia entre ejemplares, se utilizará la ecuación 2.71.

$$\frac{1}{1+d(x_i,x_j)} \qquad \text{Ec. 2.71}$$

- **Algoritmo k-estrella**

El algoritmo K* [CLTR95] es una técnica de data mining basada en ejemplares en la que la medida de la distancia entre ejemplares se basa en la teoría de la información. Una forma intuitiva de verlo es que la distancia entre dos ejemplares se define como la complejidad de transformar un ejemplar en el otro. El cálculo de la complejidad se basa en primer lugar en definir un conjunto de transformaciones $T=\{t_1, t_2, ..., t_n, \sigma\}$ para pasar de un ejemplo (valor de atributo) *a* a uno *b*. La transformación σ es la de parada y es la transformación identidad ($\sigma(a)=a$). El conjunto *P* es el conjunto de todas las posibles secuencias de transformaciones descritos en *T** que terminan en σ, y $\bar{t}(a)$ es una de estas secuencias concretas sobre el ejemplo *a*. Esta secuencia de transformaciones tendrá una probabilidad determinada $p(\bar{t})$, definiéndose la función de probabilidad *P*(b|a)* como la probabilidad de pasar del ejemplo a al ejemplo b a través de cualquier secuencia de transformaciones, tal y como se muestra en la ecuación 2.72.

$$P^*(b|a)=\sum_{\bar{t}\in P:\bar{t}(a)=b} p(\bar{t}) \qquad \text{Ec. 2.72}$$

Esta función de probabilidad cumplirá las propiedades que se muestran en 2.73.

$$\sum_b P^*(b|a)=1;\quad 0\le P^*(b|a)\le 1 \qquad \text{Ec. 2.73}$$

La función de distancia *K** se define entonces tomando logaritmos, tal y como se muestra en la ecuación 2.74.

$$K^*(b|a)=-log_2 P^*(b|a) \qquad \text{Ec. 2.74}$$

Realmente *K** no es una función de distancia dado que, por ejemplo *K*(a|a)* generalmente no será exactamente *0*, además de que el operador | no es simétrico, esto es, *K*(a|b)* no es igual que *K*(b|a)*. Sin embargo, esto no interfiere en el algoritmo K*. Además, la función *K** cumple las propiedades que se muestran en la ecuación 2.75.

$$K^*(b|a)\ge 0;\ K^*(c|b)+K^*(b|a)\ge K^*(c|a) \qquad \text{Ec. 2.75}$$

Una vez explicado cómo se obtiene la función *K** y cuales son sus propiedades, se presenta a continuación la expresión concreta de la función *P*,* de la que se obtiene *K*,* para los tipos de atributos admitidos por el algoritmo: numéricos y simbólicos.

Probabilidad de transformación para los atributos permitidos

En cuanto a los atributos numéricos, las transformaciones consideradas serán restar del valor *a* un número *n* o sumar al valor *a* un número *n*, siendo *n* un número mínimo. La probabilidad de pasar de un ejemplo con valor *a* a uno con valor *b* vendrá determinada únicamente por el valor absoluto de la diferencia entre *a* y *b*, que se denominará *x*. Se escribirá la función de probabilidad como una función de densidad, tal y como se muestra en la ecuación 2.76, donde x_0 será una medida de longitud de la escala, por ejemplo, la media esperada para *x* sobre la distribución *P**. Es necesario elegir un x_0 razonable. Posteriormente se mostrará un método para elegir este factor. Para los simbólicos, se considerarán las probabilidades de aparición de cada uno de los valores de dicho atributo.

$$P^*(x) = \frac{1}{2x_0} e^{-x/x_0}\, dx \qquad \text{Ec. 2.76}$$

Si el atributo tiene un total de *n* posibles valores, y la probabilidad de aparición del valor *i* del atributo es p_i (obtenido a partir de las apariciones en los ejemplos de entrenamiento), se define la probabilidad de transformación de un ejemplo con valor *i* a uno con valor *j* como se muestra en la ecuación 2.77.

$$P^*(j \mid i) = \begin{cases} (1 - s)p_j & si\ i \neq j \\ s + (1 - s)p_i & si\ i = j \end{cases} \qquad \text{Ec. 2.77}$$

En esta ecuación *s* es la probabilidad del símbolo de parada (σ). De esta forma, se define la probabilidad de cambiar de valor como la probabilidad de que no se pare la transformación multiplicado por la probabilidad del valor de destino, mientras la probabilidad de continuar con el mismo valor es la probabilidad del símbolo de parada más la probabilidad de que se continúe transformando multiplicado por la probabilidad del valor de destino. También es importante, al igual que con el factor x_0, definir correctamente la probabilidad *s*. Y como ya se comentó con x_0, posteriormente se comentará un método para obtenerlo. También deben tenerse en cuenta la posibilidad de los atributos con valores desconocidos. Cuando los valores desconocidos aparecen en los ejemplos de entrenamiento se propone como solución el considerar que el atributo desconocido se determina a través del resto de ejemplares de entrenamiento. Esto se muestra en la ecuación 2.78, donde *n* es el número de ejemplos de entrenamiento.

$$P^*(? \mid a) = \sum_{b=1}^{n} \frac{P^*(b \mid a)}{n} \qquad \text{Ec. 2.78}$$

Combinación de atributos

Ya se han definido las funciones de probabilidad para los tipos de atributos permitidos. Pero los ejemplos reales tienen más de un atributo, por lo que es necesario combinar los resultados obtenidos para cada atributo. Y para combinarlos, y definir así la distancia entre dos ejemplos, se entiende la probabilidad de transformación de un ejemplar en otro como la probabilidad de transformar el primer atributo del primer ejemplo en el del segundo, seguido de la transformación del segundo atributo del primer ejemplo en el del segundo, etc. De esta forma, la probabilidad de transformar

un ejemplo en otro viene determinado por la multiplicación de las probabilidades de transformación de cada atributo de forma individual, tal y como se muestra en la ecuación 2.79. En esta ecuación *m* será el número de atributo de los ejemplos. Y con esta definición la distancia entre dos ejemplos se define como la suma de distancias entre cada atributo de los ejemplos.

$$P^*(E_2 \mid E_1) = \prod_{i=1}^{m} P^*(v_{2i} \mid v_{1i}) \quad \text{Ec. 2.79}$$

Selección de los parámetros aleatorios

Para cada atributo debe determinarse el valor para los parámetros *s* o x_0 según se trate de un atributo simbólico o numérico respectivamente. Y el valor de este atributo es muy importante. Por ejemplo, si a *s* se le asigna un valor muy bajo las probabilidades de transformación serán muy altas, mientras que si s se acerca a *0* las probabilidades de transformación serán muy bajas. Y lo mismo ocurriría con el parámetro x_0. En ambos casos se puede observar cómo varía la función de probabilidad *P** según se varía el número de ejemplos incluidos partiendo desde *1* (vecino más cercano) hasta *n* (todos los ejemplares con el mismo peso). Se puede calcular para cualquier función de probabilidad el número efectivo de ejemplos como se muestra en la ecuación 2.80, en la que *n* es el número de ejemplos de entrenamiento y n_0 es el número de ejemplos con la distancia mínima al ejemplo *a* (para el atributo considerado). El algoritmo K* escogerá para x_0 (o *s*) un número entre n_0 y *n*.

$$n_0 \leq \frac{\left(\sum_{b=1}^{n} P^*(b \mid a)\right)^2}{\sum_{b=1}^{n} P^*(b \mid a)^2} \leq n \quad \text{Ec. 2.80}$$

Por conveniencia se expresa el valor escogido como un *parámetro de mezclado* [blending] *b*, que varía entre *b=0%* (n_0) y *b=100%* (*n*). La configuración de este parámetro se puede ver como una *esfera de influencia* que determina cuantos vecinos de *a* deben considerarse importantes. Para obtener el valor correcto para el parámetro x_0 (o *s*) se realiza un proceso iterativo en el que se obtienen las esferas de influencia máxima (x_0 o *s* igual a *0*) y mínima (x_0 o *s* igual a *1*), y se aproximan los valores para que dicha esfera se acerque a la necesaria para cumplir con el parámetro de mezclado.

En la figura 2.33 se presenta un ejemplo práctico de cómo obtener los valores para los parámetros x_0 o *s*. Se va a utilizar para ello el problema que se presentó en la tabla 2.1, y más concretamente el atributo *Vista* con el valor igual a *Lluvioso*, de dicho problema.

Obtención de **s** para **Vista = Lluvioso**

Objetivo:

$$esfera = \frac{b}{100} * n_i + n_{lluvioso} \quad \forall i \neq lluvioso = 0,2 * 9 + 5 = 6,8$$

Iteración 0 (Inicio):

$$vInferior_0 = 0 + EPSILON/2 = 0,005 \Rightarrow esfera = 13,99928$$

$$vSuperior_0 = 1 - EPSILON/2 = 0,995 \Rightarrow esfera = 5,03012$$

$$valor_0 = 1 - b/100 = 0,8 \Rightarrow esfera = 6,41218$$

Iteración 1:

$$vInferior_1 = vInferior_0 ; vSuperior_1 = valor_0$$

$$valor_1 = vInferior_1 + vSuperior_1 / 2 = 0,4025 \Rightarrow esfera = 10,63580$$

Iteración 2:

$$vInferior_2 = valor_1 ; vSuperior_2 = vSuperior_1$$

$$valor_2 = vInferior_2 + vSuperior_2 / 2 = 0,60125 \Rightarrow esfera = 8,29876$$

$$esfera = \frac{\left(\sum_{b=i}^{n} P^*(b|lluv)\right)^2}{\sum_{b=i}^{n} P^*(b|lluv)^2} = \frac{\left(P^*(sol|lluv)*n_{sol} + P^*(nub|lluv)*n_{nub} + P^*(lluv|lluv)*n_{lluv}\right)^2}{P^*(sol|lluv)^2 * n_{sol} + P^*(nub|lluv)^2 * n_{nub} + P^*(lluv|lluv)^2 * n_{lluv}} =$$

$$= \frac{(0,00949*5 + 0,00949*4 + 0,05244*5)^2}{0,00949^2*5 + 0,00949^2*4 + 0,05244^2*5} = \frac{0,12083}{0,01456} = 8,29876$$

$$P^*(sol|lluv) = \frac{\frac{1-s}{nv}}{m} = \frac{\frac{1-0,60125}{3}}{14} = 0,00949$$

$$P^*(nub|lluv) = \frac{\frac{1-0,60125}{3}}{14} = 0,00949$$

$$P^*(lluv|lluv) = \frac{s + \frac{1-s}{nv}}{m} = \frac{0,60125 + \frac{1-0,60125}{3}}{14} = 0,05244$$

Iteración 3:

$$vInferior_3 = valor_2 ; vSuperior_3 = vSuperior_2$$

$$valor_3 = vInferior_3 + vSuperior_3 / 2 = 0,70062 \Rightarrow esfera = 7,29193$$

...

Iteración 8:

$$vInferior_8 = vInferior_7 = 0,75031$$

$$vSuperior_8 = valor_7 = 0,75652$$

$$valor_8 = vInferior_8 + vSuperior_8 / 2 = 0,75341 \Rightarrow esfera = 6,80871$$

abs(esfera – objetivo) < EPSILON => Conseguido! => s = 0,75341

Figura 3.31: *Ejemplo de obtención del parámetros de un atributo simbólico con el algoritmo K*.*

En la figura 2.33 se muestra cómo el objetivo es conseguir un valor para *s* tal que se obtenga una esfera de influencia de *6,8* ejemplos. Los parámetros de configuración necesarios para el funcionamiento del sistema son: el parámetro de mezclado *b*, en este caso igual a *20%*; una constante denominada *EPSILON*, en este caso igual a *0,01*, que determina entre otras cosas cuándo se considera alcanzada la esfera de influencia deseada. En cuanto a la nomenclatura empleada, *n* será el número total de ejemplos de entrenamiento, *nv* el número de valores que puede adquirir el atributo, y se han empleado abreviaturas para denominar los valores del atributo: *lluv* por lluvioso, *nub* por nublado y *sol* por soleado.

Tal y como puede observarse en la figura 2.33, las ecuaciones empleadas para el cálculo de la esfera y de *P** no son exactamente las definidas en las ecuaciones definidas anteriormente. Sin embargo, en el ejemplo se han empleado las implementadas en la herramienta WEKA por los creadores del algoritmo. En cuanto al

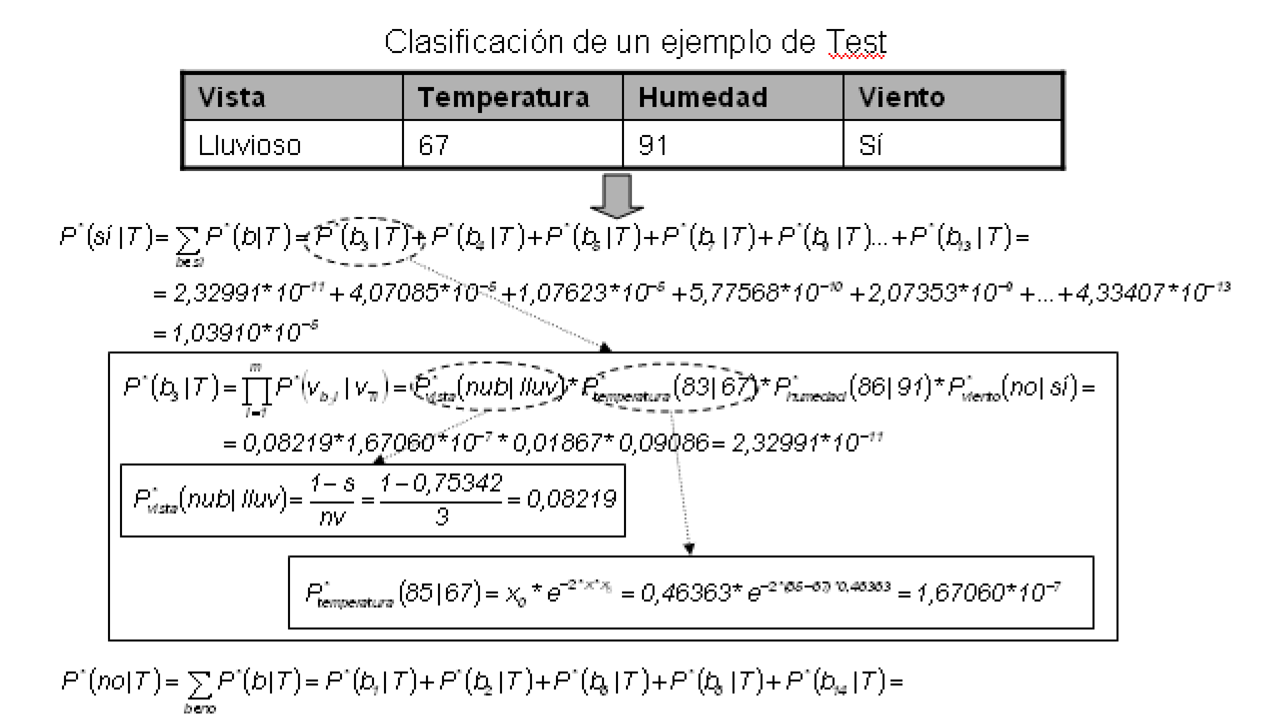
ejemplo en sí, se muestra cómo son necesarias 8 iteraciones para llegar a conseguir el objetivo planteado, siendo el resultado de dicho proceso, el valor de *s*, igual a *0,75341*.

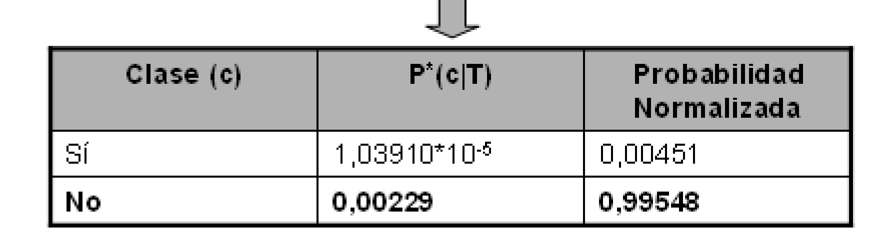
Clasificación de un ejemplo

Se calcula la probabilidad de que un ejemplo *a* pertenezca a la clase *c* sumando la probabilidad de *a* a cada ejemplo que es miembro de *c*, tal y como se muestra en 2.81.

$$P^*(c|a) = \sum_{b \in c} P^*(b|a) \qquad \text{Ec. 2.81}$$

Se calcula la probabilidad de pertenencia a cada clase y se escoge la que mayor resultado haya obtenido como predicción para el ejemplo.





Clase (c)	P*(c|T)	Probabilidad Normalizada
Sí	1,03910*10-5	0,00451
No	**0,00229**	**0,99548**

Figura 3.32: *Ejemplo de clasificación con K*.*

Una vez definido el modo en que se clasifica un determinado ejemplo de *test* mediante el algoritmo K*, en la figura 2.34 se muestra un ejemplo concreto en el que

se emplea dicho algoritmo. En el ejemplo se clasifica un ejemplo de *test* tomando como ejemplos de entrenamiento los que se mostraron en la tabla 2.1, tomando los atributos *Temperatura* y *Humedad* como numéricos. El proceso que se sigue para determinar a qué clase pertenece un ejemplo de *test* determinado es el siguiente: en primer lugar, habría que calcular los parámetros x_0 y s que aún no se conocen para los pares *atributo-valor* del ejemplo de *test*. Posteriormente se aplican las ecuaciones, que de nuevo no son exactamente las definidas anteriormente: se han empleado las que los autores del algoritmo implementan en la herramienta WEKA. Una vez obtenidas las probabilidades, se normalizan y se escoge la mayor de las obtenidas. En este caso hay más de un *99%* de probabilidad a favor de la clase *no*. Esto se debe a que el ejemplo *14* (el último) es casi idéntico al ejemplo de *test* por clasificar. En este ejemplo no se detallan todas las operaciones realizadas, sino un ejemplo de cada tipo: un ejemplo de la obtención de *P** para un atributo simbólico, otro de la obtención de *P** para un atributo numérico y otro para la obtención de la probabilidad de transformación del ejemplo de *test* en un ejemplo de entrenamiento.

3.5.6. Redes de Neuronas

Las redes de neuronas constituyen una técnica inspirada en los trabajos de investigación, iniciados en 1930, que pretendían modelar computacionalmente el aprendizaje humano llevado a cabo a través de las neuronas en el cerebro [RM86, CR95]. Posteriormente se comprobó que tales modelos no eran del todo adecuados para describir el aprendizaje humano. Las redes de neuronas constituyen una nueva forma de analizar la información con una diferencia fundamental con respecto a las técnicas tradicionales: son capaces de detectar y aprender complejos patrones y características dentro de los datos [SN88, FU94]. Se comportan de forma parecida a nuestro cerebro aprendiendo de la experiencia y del pasado, y aplicando tal conocimiento a la resolución de problemas nuevos. Este aprendizaje se obtiene como resultado del adiestramiento ("*training*") y éste permite la sencillez y la potencia de adaptación y evolución ante una realidad cambiante y muy dinámica. Una vez adiestradas las redes de neuronas pueden hacer previsiones, clasificaciones y segmentación. Presentan además, una eficiencia y fiabilidad similar a los métodos estadísticos y sistemas expertos, si no mejor, en la mayoría de los casos. En aquellos casos de muy alta complejidad las redes neuronales se muestran como especialmente útiles dada la dificultad de modelado que supone para otras técnicas. Sin embargo las redes de neuronas tienen el inconveniente de la dificultad de acceder y comprender los modelos que generan y presentan dificultades para extraer reglas de tales modelos. Otra característica es que son capaces de trabajar con datos incompletos e, incluso, contradictorios lo que, dependiendo del problema, puede resultar una ventaja o un inconveniente. Las redes de neuronas poseen las dos formas de aprendizaje: supervisado y no supervisado; ya comentadas [WI98], derivadas del tipo de paradigma que usan: el no supervisado (usa paradigmas como los ART “*Adaptive Resonance Theory*"), y el supervisado que suele usar el paradigma del “*Backpropagation*" [RHW86].

Las redes de neuronas están siendo utilizadas en distintos y variados sectores como la industria, el gobierno, el ejército, las comunicaciones, la investigación aerospacial, la banca y las finanzas, los seguros, la medicina, la distribución, la robótica, el marketing, etc. En la actualidad se está estudiando la posibilidad de utilizar técnicas avanzadas y novedosas como los Algoritmos Genéticos para crear nuevos paradigmas que mejoren el adiestramiento y la propia selección y diseño de la

arquitectura de la red (número de capas y neuronas), diseño que ahora debe realizarse en base a la experiencia del analista y para cada problema concreto.

- **Estructura de las Redes de Neuronas**

Las redes neuronales se construyen estructurando en una serie de niveles o capas (al menos tres: entrada, procesamiento u oculta y salida) compuestas por nodos o "neuronas", que tienen la estructura que se muestra en la figura 2.35.

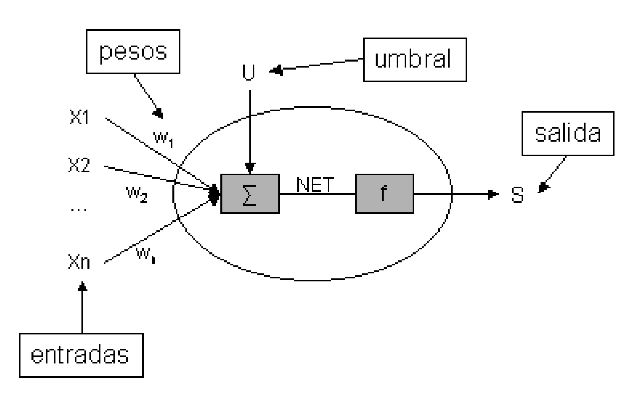


Figura 3.33: *Estructura de una neurona.*

Tanto el umbral como los pesos son constantes que se inicializarán aleatoriamente y durante el proceso de aprendizaje serán modificados. La salida de la neurona se define tal y como se muestra en las ecuaciones 2.82 y 2.83.

$$NET = \sum_{i=1}^{N} X_i w_i + U \qquad \text{Ec. 2.82}$$

$$S = f(NET) \qquad \text{Ec. 2.83}$$

Como función f se suele emplear una función sigmoidal, bien definida entre *0* y *1* (ecuación 2.84) o entre *–1* y *1* (ecuación 2.85).

$$f(x) = \frac{1}{1+e^{-x}} \qquad \text{Ec. 2.84}$$

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad \text{Ec. 2.85}$$

Cada neurona está conectada a todas las neuronas de las capas anterior y posterior a través de los pesos o "dendritas", tal y como se muestra en la figura 2.36.

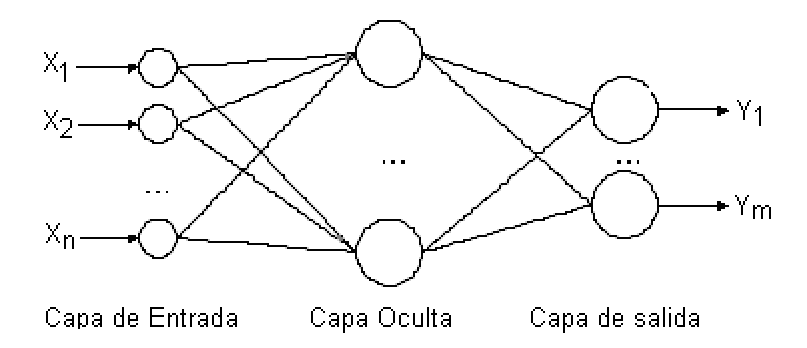


Figura 3.34: *Estructura de la red de neuronas.*

Cuando un nodo recibe las entradas o "estímulos" de otras los procesa para producir una salida que transmite a la siguiente capa de neuronas. La señal de salida tendrá una intensidad fruto de la combinación de la intensidad de las señales de entrada y de los pesos que las transmiten. Los pesos o dendritas tienen un valor distinto para cada par de neuronas que conectan pudiendo así fortalecer o debilitar la conexión o comunicación entre neuronas particulares. Los pesos son modificados durante el proceso de adiestramiento.

El diseño de la red de neuronas consistirá, entre otras cosas, en la definición del número de neuronas de las tres capas de la red. Las neuronas de la capa de entrada y las de la capa de salida vienen dadas por el problema a resolver, dependiendo de la codificación de la información. En cuanto al número de neuronas ocultas (y/o número de capas ocultas) se determinará por prueba y error. Por último, debe tenerse en cuenta que la estructura de las neuronas de la capa de entrada se simplifica, dado que su salida es igual a su entrada: no hay umbral ni función de salida.

- **Proceso de adiestramiento (retropropagación)**

Existen distintos métodos o paradigmas mediante los cuales estos pesos pueden ser variados durante el adiestramiento de los cuales el más utilizado es el de retropropagación [Backpropagation] [RHW86]. Este paradigma varía los pesos de acuerdo a las diferencias encontradas entre la salida obtenida y la que debería obtenerse. De esta forma, si las diferencias son grandes se modifica el modelo de forma importante y según van siendo menores, se va convergiendo a un modelo final estable. El error en una red de neuronas para un patrón [$x = (x_1, x_2, \ldots, x_n), t(x)$], siendo *x* el patrón de entrada, *t(x)* la salida deseada e *y(x)* la proporcionada por la red, se define como se muestra en la ecuación 2.86 para *m* neuronas de salida y como se muestra en la ecuación 2.87 para *1* neurona de salida.

$$e(x) = \|t(x) - y(x)\|^2 = \frac{1}{2}\sum_{i=1}^{m}(t_i(x) - y_i(x))^2$$ Ec. 2.86

$$e(x) = \frac{1}{2}(t(x) - y(x))^2$$ Ec. 2.87

El método de descenso de gradiente consiste en modificar los parámetros de la red siguiendo la dirección negativa del gradiente del error. Lo que se realizaría mediante 2.88.

$$w^{nuevo} = w^{anterior} + \alpha\left(-\frac{\partial e}{\partial w}\right) = w^{anterior} - \alpha\frac{\partial e}{\partial w}$$ Ec. 2.88

En la ecuación 2.88, *w* es el peso a modificar en la red de neuronas (pasando de $w^{anterior}$ a w^{nuevo}) y α es la razón de aprendizaje, que se encarga de controlar cuánto se desplazan los pesos en la dirección negativa del gradiente. Influye en la velocidad de convergencia del algoritmo, puesto que determina la magnitud del desplazamiento. El algoritmo de retropropagación es el resultado de aplicar el método de descenso del gradiente a las redes de neuronas. El algoritmo completo de retropropagación se muestra en la figura 2.37.

Paso 1: Inicialización aleatoria de los pesos y umbrales.

Paso 2: Dado un patrón del conjunto de entrenamiento *(x, t(x))*, se presenta el vector *x* a la red y se calcula la salida de la red para dicho patrón, *y(x)*.

Paso 3: Se evalúa el error *e(x)* cometido por la red.

Paso 4: Se modifican todos los parámetros de la red utilizando la ec.2.88.

Paso 5: Se repiten los pasos 2, 3 y 4 para todos los patrones de entrenamiento, completando así un ciclo de aprendizaje.

Paso 6: Se realizan n ciclos de aprendizaje (pasos 2, 3, 4 y 5) hasta que se verifique el criterio de parada establecido.

Figura 3.35: *Pseudocódigo del algoritmo de retropropagación.*

En cuanto al criterio de parada, se debe calcular la suma de los errores en los patrones de entrenamiento. Si el error es constante de un ciclo a otro, los parámetros dejan de sufrir modificaciones y se obtiene así el error mínimo. Por otro lado, también se debe tener en cuenta el error en los patrones de validación, que se presentarán a la red tras *n* ciclos de aprendizaje. Si el error en los patrones de validación evoluciona favorablemente se continúa con el proceso de aprendizaje. Si el error no desciende, se detiene el aprendizaje.

3.5.7. Lógica borrosa (“Fuzzy logic”)

La lógica borrosa surge de la necesidad de modelar la realidad de una forma más exacta evitando precisamente el determinismo o la exactitud [ZAD65, CPS98]. En palabras menos pretenciosas lo que la lógica borrosa permite es el tratamiento probabilístico de la categorización de un colectivo [ZAD65].

Así, para establecer una serie de grupos, segmentos o clases en los cuales se puedan clasificar a las personas por la edad, lo inmediato sería proponer unas edades límite para establecer tal clasificación de forma disjunta. Así los niños serían aquellos cuya edad fuera menor a los 12 años, los adolescentes aquellos entre 12 y 17 años, los jóvenes aquellos entre 18 y 35, las personas maduras entre 36 y 45 años y así sucesivamente. Se habrían creado unos grupos disjuntos cuyo tratamiento, a efectos de clasificación y procesamiento, es muy sencillo: basta comparar la edad de cada persona con los límites establecidos. Sin embargo enseguida se observa que esto supone una simplificación enorme dado que una persona de 16 años 11 meses y veinte días pertenecería al grupo de los adolescentes y, seguramente, es más parecido a una persona de 18 (miembro de otro grupo) que a uno de 12 (miembro de su grupo). Lógicamente no se puede establecer un grupo para cada año, dado que sí se reconocen grupos, y no muchos, con comportamientos y actitudes similares en función de la edad. Lo que implícitamente se esta descubriendo es que las clases existen pero que la frontera entre ellas no es clara ni disjunta sino “difusa” y que una persona puede tener aspectos de su mentalidad asociados a un grupo y otros asociados a otro grupo, es decir que implícitamente se está distribuyendo la pertenencia entre varios grupos. Cuando esto se lleva a una formalización matemática surge el concepto de distribución de posibilidad, de forma que lo que entendería como función de pertenencia a un grupo de edad serían unas curvas de posibilidad. Por tanto, la lógica borrosa es aquella técnica que permite y trata la existencia de barreras difusas o suaves entre los distintos grupos en los que se categoriza un colectivo o entre los distintos elementos, factores o proporciones que concurren en una situación o solución [BS97].

Para identificar las áreas de utilización de la lógica difusa basta con determinar cuantos problemas hacen uso de la categorización disjunta en el tratamiento de los datos para observar la cantidad de posibles aplicaciones que esta técnica puede tener [ZAD65].. Sin embargo, el tratamiento ortodoxo y purista no siempre está justificado dada la complejidad que induce en el procesamiento (pasamos de valores a funciones de posibilidad) y un modelado sencillo puede ser más que suficiente. Aún así, existen problemáticas donde este modelado sí resulta justificado, como en el control de procesos y la robótica, entre otros. Tal es así que un país como Japón, líder en la industria y la automatización, dispone del "Laboratory for International Fuzzy Engineering Research" (LIFE) y empresas como Yamaichi Securities y Canon hacen un extenso uso de esta técnica.

3.5.8. Técnicas Genéticas: Algoritmos Genéticos (“Genetic Algorithms”)

Los Algoritmos Genéticos son otra técnica que tiene su inspiración, en la Biología como las Redes de Neuronas [GOLD89, MIC92, MITC96]. Estos algoritmos

representan el modelado matemático de como los cromosomas en un marco evolucionista alcanzan la estructura y composición más óptima en aras de la supervivencia. Entendiendo la evolución como un proceso de búsqueda y optimización de la adaptación de las especies que se plasma en mutaciones y cambios de los genes o cromosomas, los Algoritmos Genéticos hacen uso de las técnicas biológicas de reproducción (mutación y cruce) para ser utilizadas en todo tipo de problemas de búsqueda y optimización. Se da la mutación cuando alguno o algunos de los genes cambian bien de forma aleatoria o de forma controlada vía funciones y se obtiene el cruce cuando se construye una nueva solución a partir de dos contribuciones procedentes de otras soluciones "padre". En cualquier caso, tales transformaciones se realizan sobre aquellos especimenes o soluciones más aptas o mejor adaptadas. Dado que los mecanismos biológicos de evolución han dado lugar a soluciones, los seres vivos, realmente idóneas cabe esperar que la aplicación de tales mecanismos a la búsqueda y optimización de otro tipo de problemas tenga el mismo resultado. De esta forma los Algoritmos Genéticos transforman los problemas de búsqueda y optimización de soluciones un proceso de evolución de unas soluciones de partida. Las soluciones se convierten en cromosomas, transformación que se realiza pasando los datos a formato binario, y a los mejores se les van aplicando las reglas de evolución (funciones probabilísticas de transición) hasta encontrar la solución óptima. En muchos casos, estos mecanismos brindan posibilidades de convergencia más rápidos que otras técnicas.

El uso de estos algoritmos no está tan extendido como otras técnicas, pero van siendo cada vez más utilizados directamente en la solución de problemas, así como en la mejora de ciertos procesos presentes en otras herramientas. Así, por ejemplo, se usan para mejorar los procesos de adiestramiento y selección de arquitectura de las redes de neuronas, para la generación e inducción de árboles de decisión y para la síntesis de programas a partir de ejemplos ("Genetic Programming").

Capítulo 4. Técnicas de Análisis de Datos en Weka

Introducción

En este capítulo se presenta de forma concisa y práctica la herramienta de minería de datos WEKA. WEKA, acrónimo de *Waikato Environment for Knowledge Analysis*, es un entorno para experimentación de análisis de datos que permite aplicar, analizar y evaluar las técnicas más relevantes de análisis de datos, principalmente las provenientes del aprendizaje automático, sobre cualquier conjunto de datos del usuario. Para ello únicamente se requiere que los datos a analizar se almacenen con un cierto formato, conocido como *ARFF* (*Attribute-Relation File Format*).

WEKA se distribuye como software de libre distribución desarrollado en Java. Está constituido por una serie de paquetes de código abierto con diferentes técnicas de preprocesado, clasificación, agrupamiento, asociación, y visualización, así como facilidades para su aplicación y análisis de prestaciones cuando son aplicadas a los datos de entrada seleccionados. Estos paquetes pueden ser integrados en cualquier proyecto de análisis de datos, e incluso pueden extenderse con contribuciones de los usuarios que desarrollen nuevos algoritmos. Con objeto de facilitar su uso por un mayor número de usuarios, WEKA además incluye una interfaz gráfica de usuario para acceder y configurar las diferentes herramientas integradas.

Este capítulo tiene un enfoque práctico y funcional, pretendiendo servir de guía de utilización de esta herramienta desde su interfaz gráfica, como material complementario a la escasa documentación disponible. Para ello se obviarán los detalles técnicos y específicos de los diferentes algoritmos, que se presentan en un capítulo aparte, y se centrará en su aplicación, configuración y análisis dentro de la herramienta. Por tanto, se remite al lector al capítulo con los detalles de los algoritmos para conocer sus características, parámetros de configuración, etc. Aquí se han seleccionado algunas de las técnicas disponibles para aplicarlas a ejemplos concretos, siguiendo el acceso desde la herramienta al resto de técnicas implementadas, una mecánica totalmente análoga a la presentada a modo ilustrativo.

Para reforzar el carácter práctico de este capítulo, además se adoptará un formato de tipo tutorial, con un conjunto de datos disponibles sobre el que se irán aplicando las diferentes facilidades de WEKA. Se sugiere que el lector

aplique los pasos indicados y realice los análisis sugeridos para cada técnica con objeto de familiarizarse y mejorar su comprensión. Los ejemplos seleccionados son contienen datos provenientes del campo de la enseñanza, correspondientes a alumnos que realizaron las pruebas de selectividad en los años 1993-2003 procedentes de diferentes centros de enseñanza secundaria de la comunidad de Madrid. Por tanto, esta guía ilustra la aplicación y análisis de técnicas de extracción de conocimiento sobre datos del campo de la enseñanza, aunque sería directa su traslación a cualquier otra disciplina.

Preparación de los datos

Los datos de entrada a la herramienta, sobre los que operarán las técnicas implementadas, deben estar codificados en un formato específico, denominado *Attribute-Relation File Format* (extensión "arff"). La herramienta permite cargar los datos en tres soportes: fichero de texto, acceso a una base de datos y acceso a través de internet sobre una dirección URL de un servidor web. En nuestro caso trabajaremos con ficheros de texto. Los datos deben estar dispuestos en el fichero de la forma siguiente: cada instancia en una fila, y con los atributos separados por comas. El formato de un fichero arff sigue la estructura siguiente:

```
% comentarios
@relation NOMBRE_RELACION
@attribute r1 real
@attribute r2 real ...
...
@attribute i1 integer
@attribute i2 integer
…
@attribute s1 {v1_s1, v2_s1,…vn_s1}
@attribute s2 {v1_s1, v2_s1,…vn_s1}
…
@data
DATOS
```

por tanto, los atributos pueden ser principalmente de dos tipos: numéricos de tipo real o entero (indicado con las palabra *real* o *integer* tras el nombre del atributo), y simbólicos, en cuyo caso se especifican los valores posibles que puede tomar entre llaves.

Muestra de datos

El fichero de datos objeto de análisis en esta guía contiene muestras correspondientes a 18802 alumnos presentados a las pruebas de selectividad y los resultados obtenidos en las pruebas. Los datos que describen cada alumno contienen la siguiente información: año, convocatoria, localidad del centro, opción cursada (de 5 posibles), calificaciones parciales obtenidas en lengua,

historia, idioma y las tres asignaturas opcionales, así como la designación de las asignaturas de idioma y las 3 opcionales cursadas, calificación en el bachillerato, calificación final y si el alumno se presentó o no a la prueba. Por tanto, puede comprobarse que la cabecera del fichero de datos, "selectividad.arff", sigue el formato mencionado anteriormente:

```
@relation selectividad

@attribute Año_académico real
@attribute convocatoria {J, S}
@attribute localidad {ALPEDRETE, ARANJUEZ, ... }
@attribute opcion1ª {1,2,3,4,5}
@attribute nota_Lengua real
@attribute nota_Historia real
@attribute nota_Idioma real
@attribute des_Idioma {INGLES, FRANCES, ALEMAN}
@attribute des_asig1 {BIOLOGIA, DIB.ARTISTICO_II,... }
@attribute calif_asig1 real
@attribute des_asig2 {BIOLOGIA, C.TIERRA, ...}
@attribute calif_asig2 real
@attribute des_asig3 {BIOLOGIA, C.TIERRA, ...}
@attribute calif_asig3 real
@attribute cal_prueba real
@attribute nota_bachi real
@attribute cal_final real
@attribute Presentado {SI, NO}
@data
...
```

Objetivos del análisis

Antes de comenzar con la aplicación de las técnicas de WEKA a los datos de este dominio, es muy conveniente hacer una consideración acerca de los objetivos perseguidos en el análisis. Como se mencionó en la introducción, un paso previo a la búsqueda de relaciones y modelos subyacentes en los datos ha de ser la comprensión del dominio de aplicación y establecer una idea clara acerca de los objetivos del usuario final. De esta manera, el proceso de análisis de datos (proceso *KDD*), permitirá dirigir la búsqueda y hacer refinamientos, con una interpretación adecuada de los resultados generados. Los objetivos, utilidad, aplicaciones, etc., del análisis efectuado no "emergen" de los datos, sino que deben ser considerados con detenimiento como primer paso del estudio.

En nuestro caso, uno de los objetivos perseguidos podría ser el intentar relacionar los resultados obtenidos en las pruebas con características o perfiles de los alumnos, si bien la descripción disponible no es muy rica y habrá que atenerse a lo que está disponible. Algunas de las preguntas que podemos plantearnos a responder como objetivos del análisis podrían ser las siguientes:

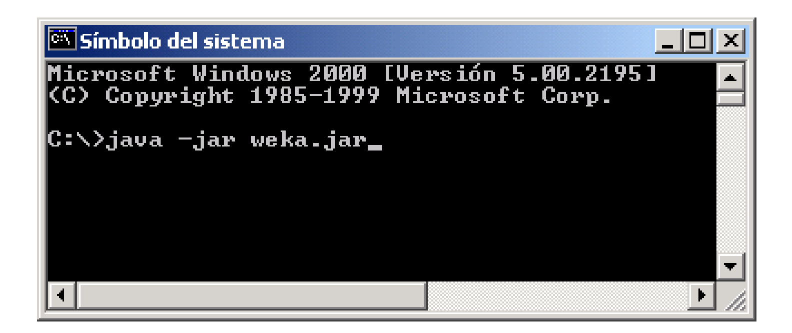
- ¿Qué caracterísitcas comunes tienen los alumnos que superan la prueba? ¿y los alumnos mejor preparados que la superan sin perjudicar su expediente?
- ¿existen grupos de alumnos, no conocidos de antemano, con características similares?
- ¿hay diferencias significativas en los resultados obtenidos según las opciones, localidades, años, etc.?,
- ¿la opción seleccionada y el resultado está influida depende del entorno?
- ¿se puede predecir la calificación del alumno con alguna variable conocida?
- ¿qué relaciones entre variables son las más significativas?

Como veremos, muchas veces el resultado alcanzado puede ser encontrar relaciones triviales o conocidas previamente, o puede ocurrir que el hecho de no encontrar relaciones significativas, lo puede ser muy relevante. Por ejemplo, saber después de un análisis exhaustivo que la opción o localidad no condiciona significativamente la calificación, o que la prueba es homogénea a lo largo de los años, puede ser una conclusión valiosa, y en este caso "tranquilizadora".

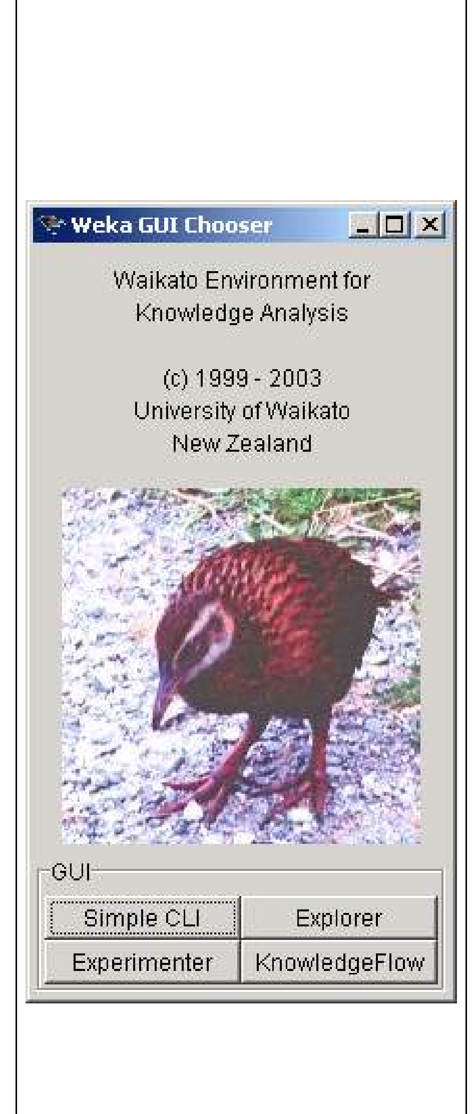
Por otra parte, este análisis tiene un enfoque introductorio e ilustrativo para acercarse a las técnicas disponibles y su manipulación desde la herramienta, dejando abierto para el investigador llevar el estudio de este dominio a resultados y conclusiones más elaboradas.

Ejecución de WEKA

WEKA se distribuye como un fichero ejecutable comprimido de java (fichero "jar"), que se invoca directamente sobre la máquina virtual JVM. En las primeras versiones de WEKA se requería la máquina virtural Java 1.2 para invocar a la interfaz gráfica, desarrollada con el paquete gráfico de Java *Swing*. En el caso de la último versión, WEKA 3-4, que es la que se ha utilizado para confeccionar estas notas, se requiere Java 1.3 o superior. La herramienta se invoca desde el intérprete de Java, en el caso de utilizar un entorno windows, bastaría una ventana de comandos para invocar al intéprete Java:



Una vez invocada, aparece la ventana de entrada a la interfaz gráfica (*GUI-Chooser*), que nos ofrece cuatro opciones posibles de trabajo:



- **Simple CLI**: la interfaz "Command-Line Interfaz" es simplemente una ventana de comandos java para ejecutar las clases de WEKA. La primera distribución de WEKA no disponía de interfaz gráfica y las clases de sus paquetes se podían ejecutar desde la línea de comandos pasando los argumentos adecuados.

- **Explorer**: es la opción que permite llevar a cabo la ejecución de los algoritmos de análisis implementados sobre los ficheros de entrada, una ejecución independiente por cada prueba. Esta es la opción sobre la que se centra la totalidad de esta guía.

- **Experimenter**: esta opción permite definir experimentos más complejos, con objeto de ejecutar uno o varios algoritmos sobre uno o varios conjuntos de datos de entrada, y comparar estadísticamente los resultados

- **KnowledgeFlow**: esta opción es una novedad de WEKA 3-4 que permite llevar a cabo las mismas acciones del "Explorer", con una configuración totalmente gráfica, inspirada en herramientas de tipo "data-flow" para seleccionar componentes y conectarlos en un proyecto de minería de datos, desde que se cargan los datos, se aplican algoritmos de tratmiento y análisis, hasta el tipo de evaluación deseada.

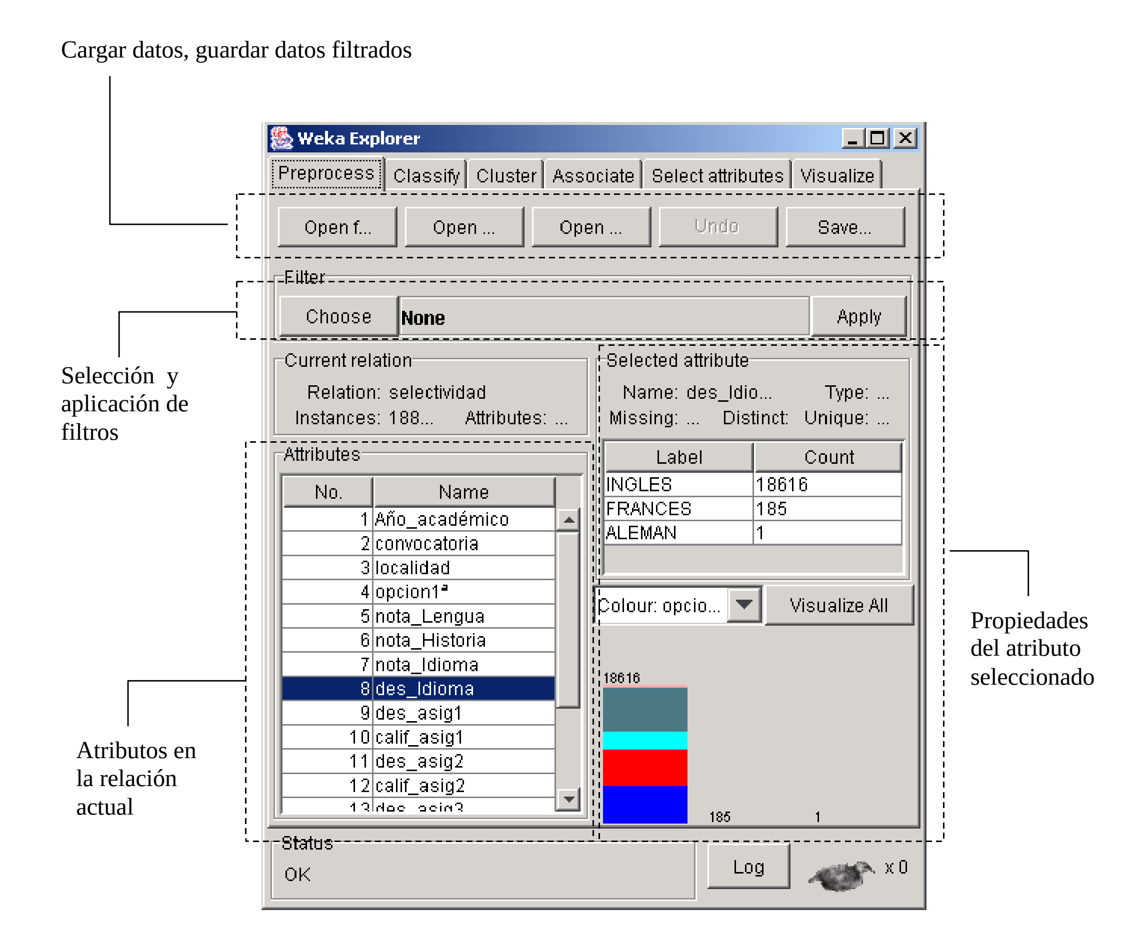
En esta guía nos centraremos únicamente en la segunda opción, *Explorer*. Una vez seleccionada, se crea una ventana con 6 pestañas en la parte superior que se corresponden con diferentes tipos de operaciones, en etapas independientes, que se pueden realizar sobre los datos:

- **Preprocess**: seleccion de la fuente de datos y preparación (filtrado).
- **Clasify**: Facilidades para aplicar esquemas de clasificación, entrenar modelos y evaluar su precisión
- **Cluster**: Algoritmos de agrupamiento
- **Associate**: Algoritmos de búsqueda de reglas de asociación
- **Select Attributes**: Búsqueda supervisada de subconjuntos de atributos representativos
- **Visualize**: Herramienta interactiva de presentación gráfica en 2D.

Además de estas pestañas de selección, en la parte inferior de la ventana aparecen dos elementos comunes. Uno es el botón de "**Log**", que al activarlo presenta una ventana textual donde se indica la secuencia de todas las operaciones que se han llevado a cabo dentro del "Explorer", sus tiempos de inicio y fin, así como los mensajes de error más frecuentes. Junto al botón de log aparece un icono de actividad (el pájaro WEKA, que se mueve cuando se está realizando alguna tarea) y un indicador de status, que indica qué tarea se está realizando en este momento dentro del Explorer.

Preprocesado de los datos

Esta es la parte primera por la que se debe pasar antes de realizar ninguna otra operación, ya que se precisan datos para poder llevar a cabo cualquier análisis. La disposición de la parte de preprocesado del *Explorer,* **Preprocess**, es la que se indica en la figura siguiente.

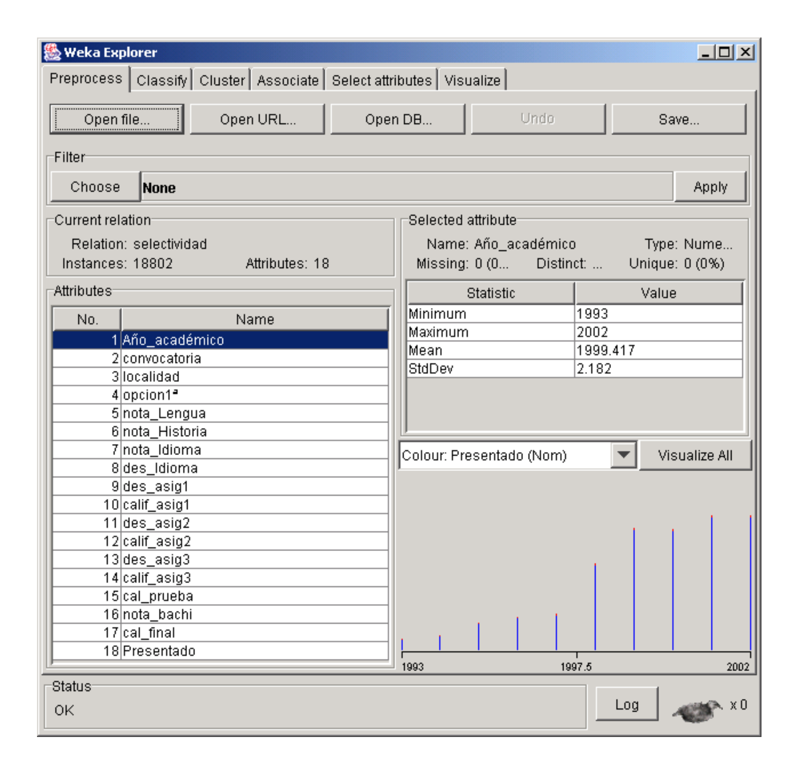


Como se indicó anteriormente, hay tres posibilidades para obtener los datos: un fichero de texto, una dirección URL o una base de datos, dadas por las opciones: **Open file**, **Open URL** y **Open DB**. En nuestro caso utilizaremos siempre los datos almacenados en un fichero, que es lo más rápido y cómodo de utilizar. La preparación del fichero de datos en formato ARFF ya se describió en la sección 1.2.

En el ejemplo que nos ocupa, abra el fichero “selectividad.arff” con la opción **Open File**.

Características de los atributos

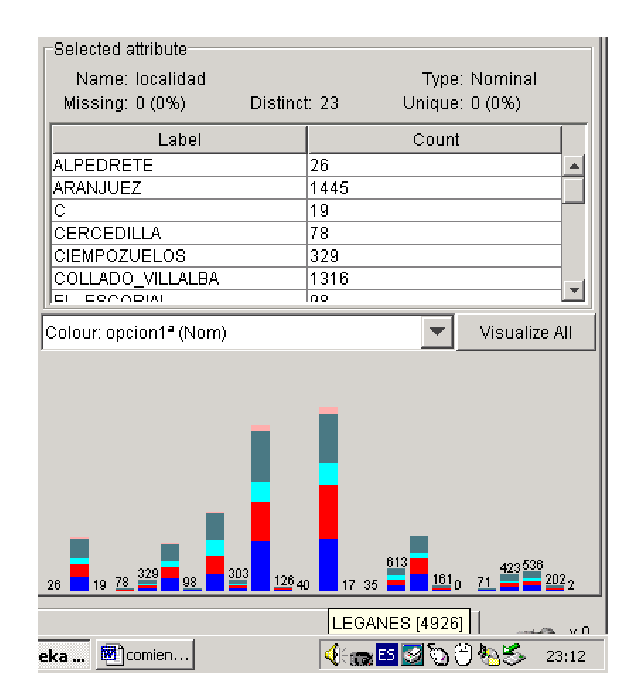
Una vez cargados los datos, aparece un cuadro resumen, *Current relation*, con el nombre de la relación que se indica en el fichero (en la línea @relation del fichero arff), el número de instancias y el número de atributos. Más abajo, aparecen listados todos los atributos disponibles, con los nombres especificados en el fichero, de modo que se pueden seleccionar para ver sus detalles y propiedades.



En la parte derecha aparecen las propiedades del atributo seleccionado. Si es un atributo simbólico, se presenta la distribución de valores de ese atributo (número de instancias que tienen cada uno de los valores). Si es numérico aparece los valores máximo, mínimo, valor medio y desviación estándar. Otras características que se destacan del atributo seleccionado son el tipo (*Type*), número de valores distintos (*Distinct*), número y porcentaje de instancias con valor desconocido para el atributo (*Missing*, codificado en el fichero arff con "?"), y valores de atributo que solamente se dan en una instancia (*Unique*).

Además, en la parte inferior se presenta gráficamente el histograma con los valores que toma el atributo. Si es simbólico, la distribución de frecuencia de los valores, si es numérico, un histograma con intervalos uniformes. En el histograma se puede presentar además con colores distintos la distribución de un segundo atributo para cada valor del atributo visualizado. Por último, hay un botón que permite visualizar los histogramas de todos los atributos simultáneamente.

A modo de ejemplo, a continuación mostramos el histograma por localidades, indicando con colores la distribuciones por opciones elegidas.



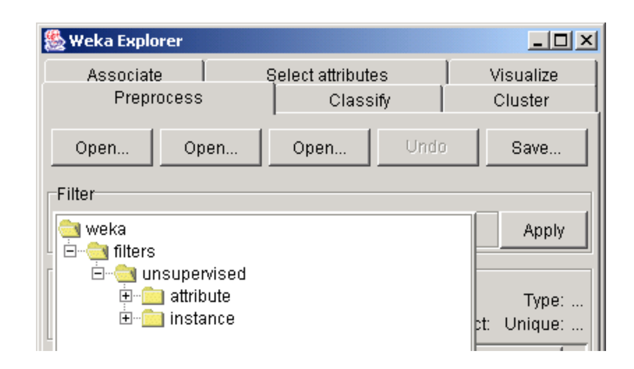
Se ha seleccionado la columna de la localidad de Leganés, la que tiene más instancias, y donde puede verse que la proporción de las opciones científicas (1 y 2) es superior a otras localidades, como Getafe, la segunda localidad en número de alumnos presentados.

Visualice a continuación los histogramas de las calificaciones de bachillerato y calificación final de la prueba, indicando como segundo atributo la convocatoria en la que se presentan los alumnos.

Trabajo con Filtros. Preparación de ficheros de muestra

WEKA tiene integrados filtros que permiten realizar manipulaciones sobre los datos en dos niveles: atributos e instancias. Las operaciones de filtrado pueden aplicarse “en cascada”, de manera que cada filtro toma como entrada el conjunto de datos resultante de haber aplicado un filtro anterior. Una vez que se ha aplicado un filtro, la relación cambia ya para el resto de operaciones llevadas a cabo en el *Experimenter*, existiendo siempre la opción de deshacer la última operación de filtrado aplicada con el botón **Undo**. Además, pueden guardarse los resultados de aplicar filtros en nuevos ficheros, que también serán de tipo ARFF, para manipulaciones posteriores.

Para aplicar un filtro a los datos, se selecciona con el botón **Choose** de **Filter**, desplegándose el árbol con todos los que están integrados.



Puede verse que los filtros de esta opción son de tipo no supervisado (*unsupervised*): son operaciones independientes del algoritmo análisis posterior, a diferencia de los filtros supervisados que se verán en la sección 1.9 de “selección de atributos”, que operan en conjunción con algoritmos de clasificación para analizar su efecto. Están agrupados según modifiquen los atributos resultantes o seleccionen un subconjunto de instancias (los filtros de atributos pueden verse como filtros "verticales" sobre la tabla de datos, y los filtros de instancias como filtros "horizontales"). Como puede verse, hay más de 30 posibilidades, de las que destacaremos únicamente algunas de las más frecuentes.

Filtros de atributos

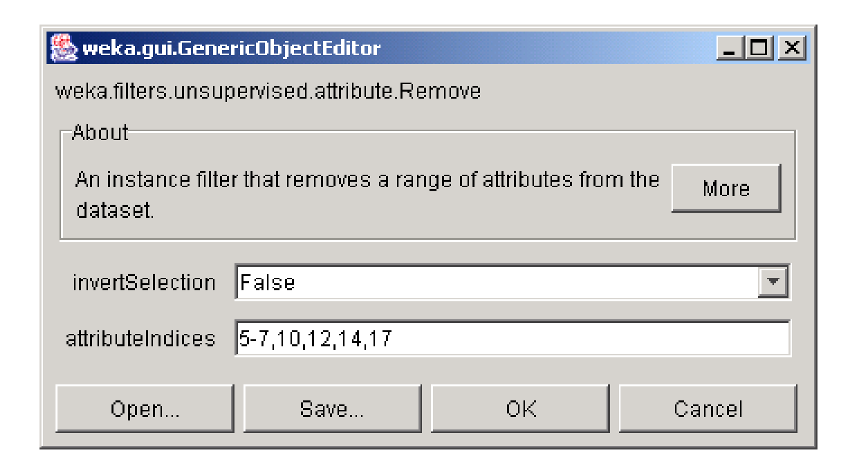
Vamos a indicar, de entre todas las posibilidades implementadas, la utilización de filtros para eliminar atributos, para discretizar atributos numéricos, y para añadir nuevos atributos con expresiones, por la frecuencia con la que se realizan estas operaciones.

Filtros de selección

Vamos a utilizar el filtro de atributos “*Remove*”, que permite eliminar una serie de atributos del conjunto de entrada. En primer lugar procedemos a seleccionarlo desde el árbol desplegado con el botón **Choose** de los filtros. A continuación lo configuraremos para determinar qué atributos queremos filtrar.

La configuración de un filtro sigue el esquema general de configuración de cualquier algoritmo integrado en WEKA. Una vez seleccionado el filtro específico con el botón **Choose**, aparece su nombre dentro del área de filtro (el lugar donde antes aparecía la palabra **None**). Se puede configurar sus parámetros haciendo clic sobre esta área, momento en el que aparece la ventana de configuración correspondiente a ese filtro particular. Si no se realiza esta operación se utilizarían los valores por defecto del filtro seleccionado.

Como primer filtro de selección, vamos a eliminar de los atributos de entrada todas las calificaciones parciales de la prueba y la calificación final, quedando como únicas calificaciones la nota de bachillerato y la calificación de la prueba. Por tanto tenemos que seleccionar los índices 5,6,7,10,12,14 y 17, indicándolo en el cuadro de configuración del filtro *Remove*:



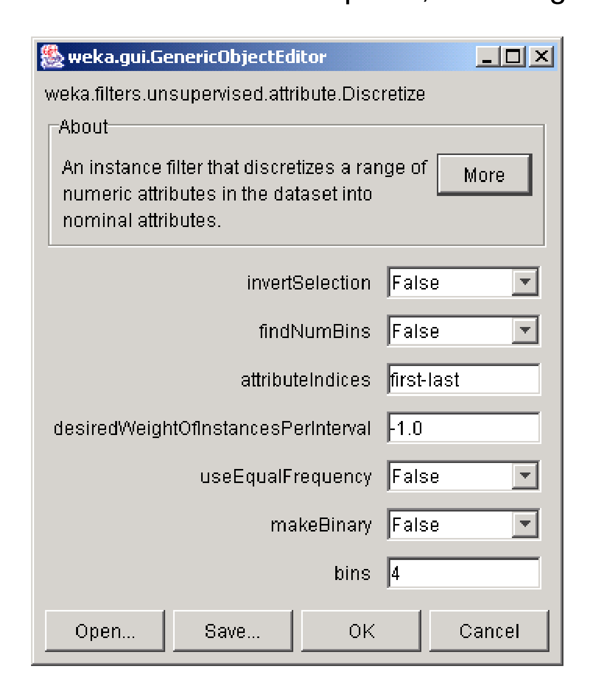
Como puede verse, en el conjunto de atributos a eliminar se pueden poner series de valores contiguos delimitados por guión (5-7) o valores sueltos entre comas (10,12,14,17). Además, puede usarse “first” y “last” para indicar el primer y último atributo, respectivamente. La opción **invertSelection** es útil cuando realmente queremos seleccionar un pequeño subconjunto de todos los atributos y eliminar el resto. **Open** y **Save** nos permiten guardar configuraciones de interés en archivos. El boton **More**, que aparece opcionalmente en algunos elementos de WEKA, muestra información de utilidad acerca de la configuración de los mismos. Estas convenciones para designar y seleccionar atributos, ayuda, y para guardar y cargar configuraciones específicas es común a otros elementos de WEKA.

Una vez configurado, al accionar el botón **Apply** del área de filtros se modifica el conjunto de datos (se filtra) y se genera una relación transformada. Esto se hace indicar en la descripción “Current Relation”, que pasa a ser la resultante de aplicar la operación correspondiente (esta información se puede ver con más nitidez en la ventana de log, que además nos indicará la cascada de filtros aplicados a la relación operativa). La relación transformada tras aplicar el filtro podría almacenarse en un nuevo fichero ARFF con el botón **Save**, dentro de la ventana **Preprocess.**

Filtros de discretización

Estos filtros son muy útiles cuando se trabaja con atributos numéricos, puesto que muchas herramientas de análisis requieren datos simbólicos, y por tanto se necesita aplicar esta transformación antes. También son necesarios cuando queremos hacer una clasificación sobre un atributo numérico, por ejemplo clasificar los alumnos aprobados y suspensos. Este filtrado transforma los atributos numéricos seleccionados en atributos simbólicos, con una serie de etiquetas resultantes de dividir la amplitud total del atributo en intervalos, con diferentes opciones para seleccionar los límites. Por defecto, se divide la amplitud del intervalo en tantas "cajas" como se indique en **bins** (por defecto 10), todas ellas de la misma amplitud.

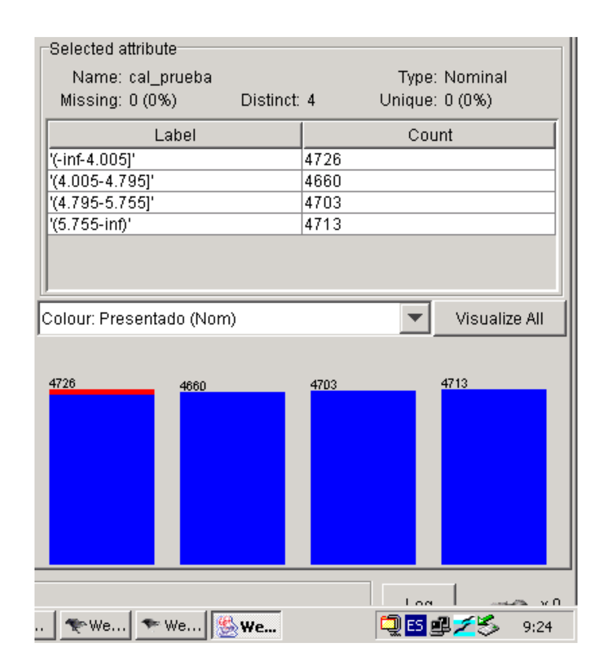
Por ejemplo, para discretizar las calificaciones numéricas en 4 categorías, todas de la misma amplitud, se configuraría así:



observe el resultado después de aplicar el filtro y los límites elegidos para cada atributo. En este caso se ha aplicado a todos los atributos numéricos con la misma configuración (los atributos seleccionados son first-last, no considerando los atributos que antes del filtrado no eran numéricos). Observe que la relación de trabajo ahora ("current relation") ahora es el resultado de aplicar en secuencia el filtro anterior y el actual.

A veces es más útil no fijar todas las cajas de la misma anchura sino forzar a una distribución uniforme de instancias por categoría, con la opción **useEqualFrequency**. La opción **findNumBins** permite opimizar el número de cajas (de la misma amplitud), con un criterio de clasificación de mínimo error en función de las etiquetas.

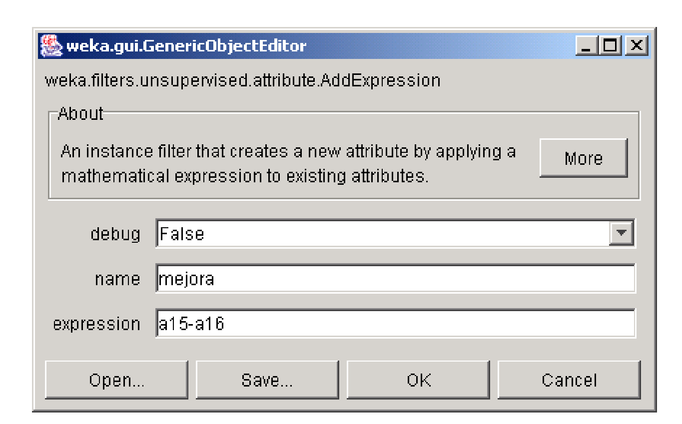
Haga una nueva discretización de la relación (eliminando el efecto del filtro anterior y dejando la relación original con el botón **Undo**) que divida las calificaciones en 4 intervalos de la misma frecuencia, lo que permite determinar los cuatro cuartiles (intervalos al 25%) de la calificación en la prueba: los intervalos delimitados por los valores {4, 4.8, 5.76}



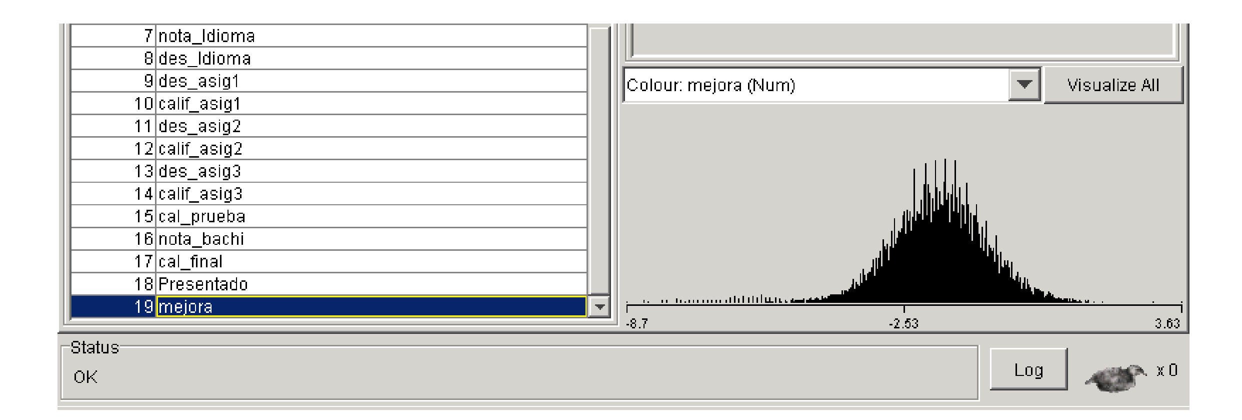
podemos ver que el 75% alcanza la nota de compensación (4). El 50% está entre 4 y 5.755, y el 25% restante a partir de 5.755.

Filtros de añadir expresiones

Muchas veces es interesante incluir nuevos atributos resultantes de aplicar expresiones a los existentes, lo que puede traer información de interés o formular cuestiones interesantes sobre los datos. Por ejemplo, vamos a añadir como atributo de interés la "mejora" sobre la nota de bachillerato, lo que puede servir para calificar el "éxito" en la prueba. Seleccionamos el filtro de atributos **AddExpression**, configurado para obtener la diferencia entre los atributos calificación en la prueba, y nota de bachillerato, en las posiciones15 y 16:



después de aplicarlo aparece este atributo en la relación, sería el número 19, con el histograma indicado en la figura:

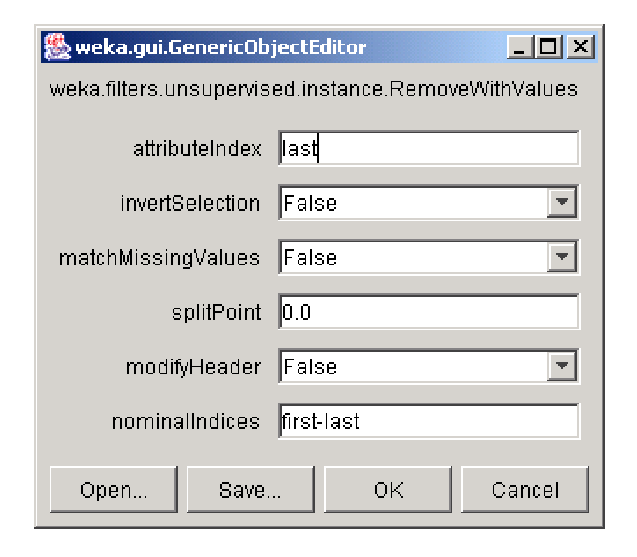


Filtros de instancias

De entre todas las posibilidades implementadas para filtros de selección de instancias (selección de rangos, muestreos, etc.), nos centraremos en la utilización de filtros para seleccionar instancias cuyos atributos cumplen determinadas condiciones.

Selección de instancias con condiciones sobre atributos

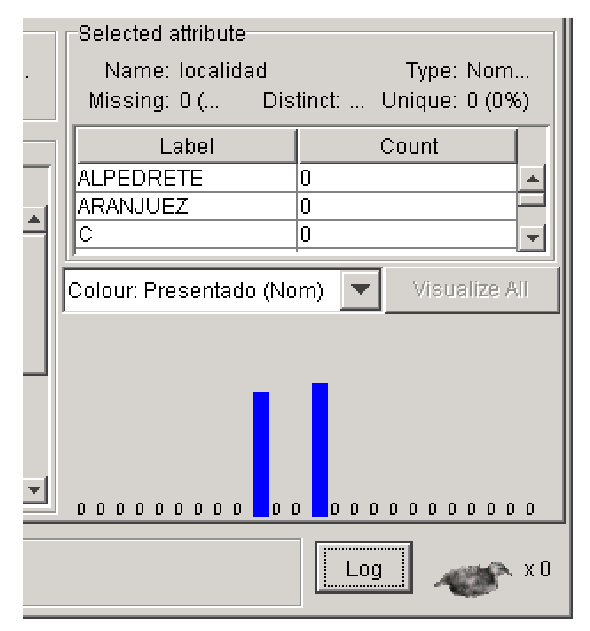
Vamos a utilizar el filtro **RemoveWithValues**, que elimina las instancias de acuerdo a condiciones definidas sobre uno de los atributos. Las opciones que aparecen en la ventana de configuración son las indicadas a continuación.

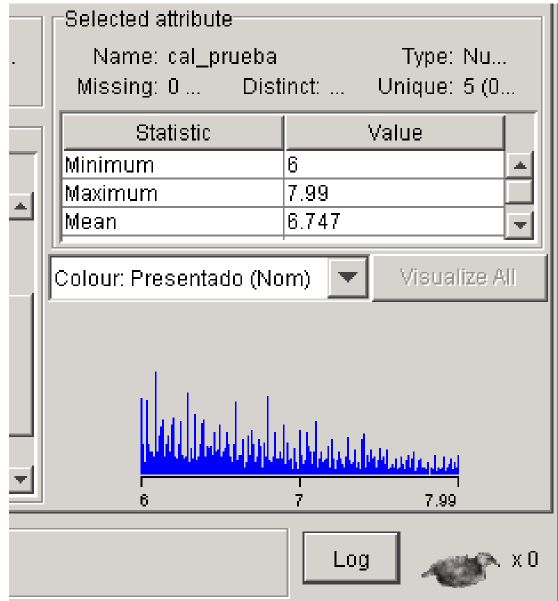


el atributo utilizado para filtrar se indica en "attributeIndex". Si es un atributo nominal, se indican los valores a filtrar en el último parámetro, "nominalIndices". Si es numérico, se filtran las instancias con un valor inferior al punto de corte, "splitPoint". Se puede invertir el criterio de filtrado mediante el campo "invertSelection".

Este filtro permite verificar una condición simple sobre un atributo. Sin embargo, es posible hacer un filtrado más complejo sobre varias condiciones aplicadas a uno o varios atributos sin más que aplicar en cascada varios filtros

A modo de ejemplo, utilice tres filtros de este tipo para seleccionar los alumnos de Getafe y Leganés con una calificación de la prueba entre 6.0 y 8.0. Compruebe el efecto de filtrado visualizando los histogramas de los atributos correspondientes (localidad y calificación en la prueba), tal y como se indica en la figura siguiente:



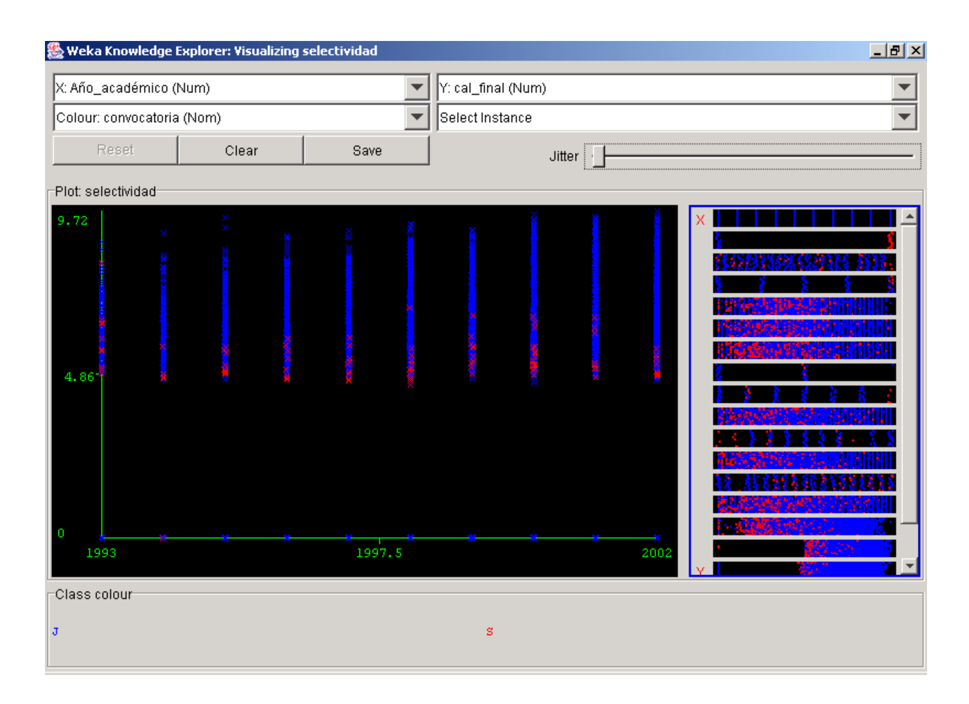


Visualización

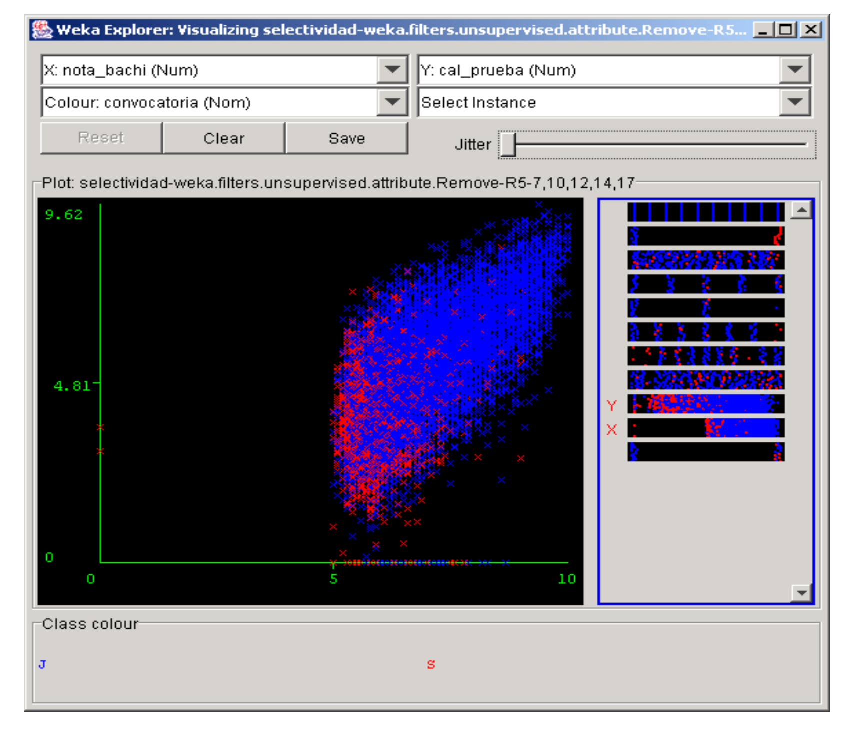
Una de las primeras etapas del análisis de datos puede ser el mero análisis visual de éstos, en ocasiones de gran utilidad para desvelar relaciones de interés utilizando nuestra capacidad para comprender imágenes. La herramienta de visualización de WEKA permite presentar gráficas 2D que relacionen pares de atributos, con la opción de utilizar además los colores para añadir información de un tercer atributo. Además, tiene incorporada una facilidad interactiva para seleccionar instancias con el ratón.

Representación 2D de los datos

Las instancias se pueden visualizar en gráficas 2D que relacionen pares de atributos. Al seleccionar la opción **Visualize** del *Explorer* aparecen todas los pares posibles de atributos en las coordenadas horizontal y vertical. La idea es que se selecciona la gráfica deseada para verla en detalle en una ventana nueva. En nuestro caso, aparecerán todas las combinaciones posibles de atributos. Como primer ejemplo vamos a visualizar el rango de calificaciones finales de los alumnos a lo largo de los años, poniendo la convocatoria (junio o septiembre) como color de la gráfica.



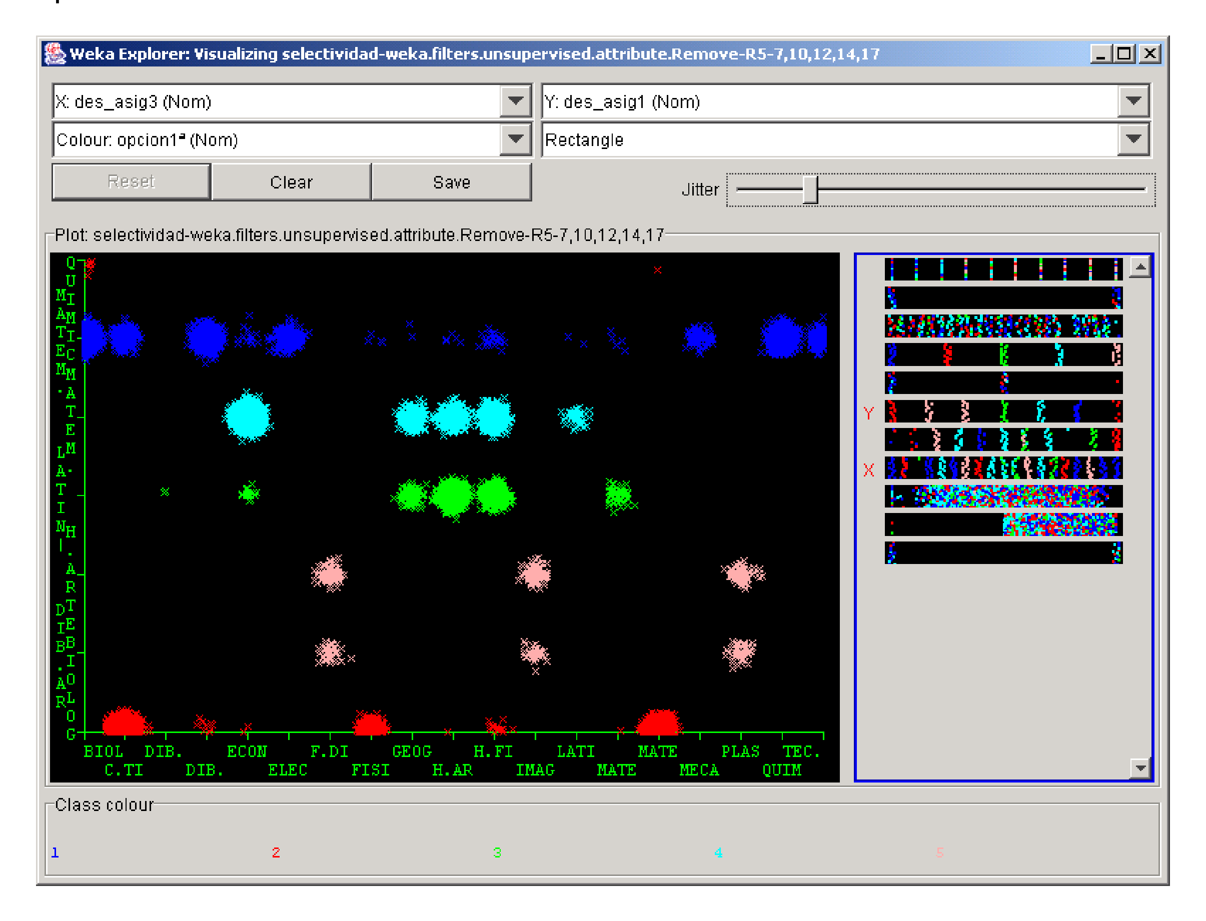
Vamos a visualizar ahora dos variables cuya relación es de gran interés, la calificación de la prueba en función de la nota de bachillerato, y tomando como color la convocatoria (junio o septiembre).

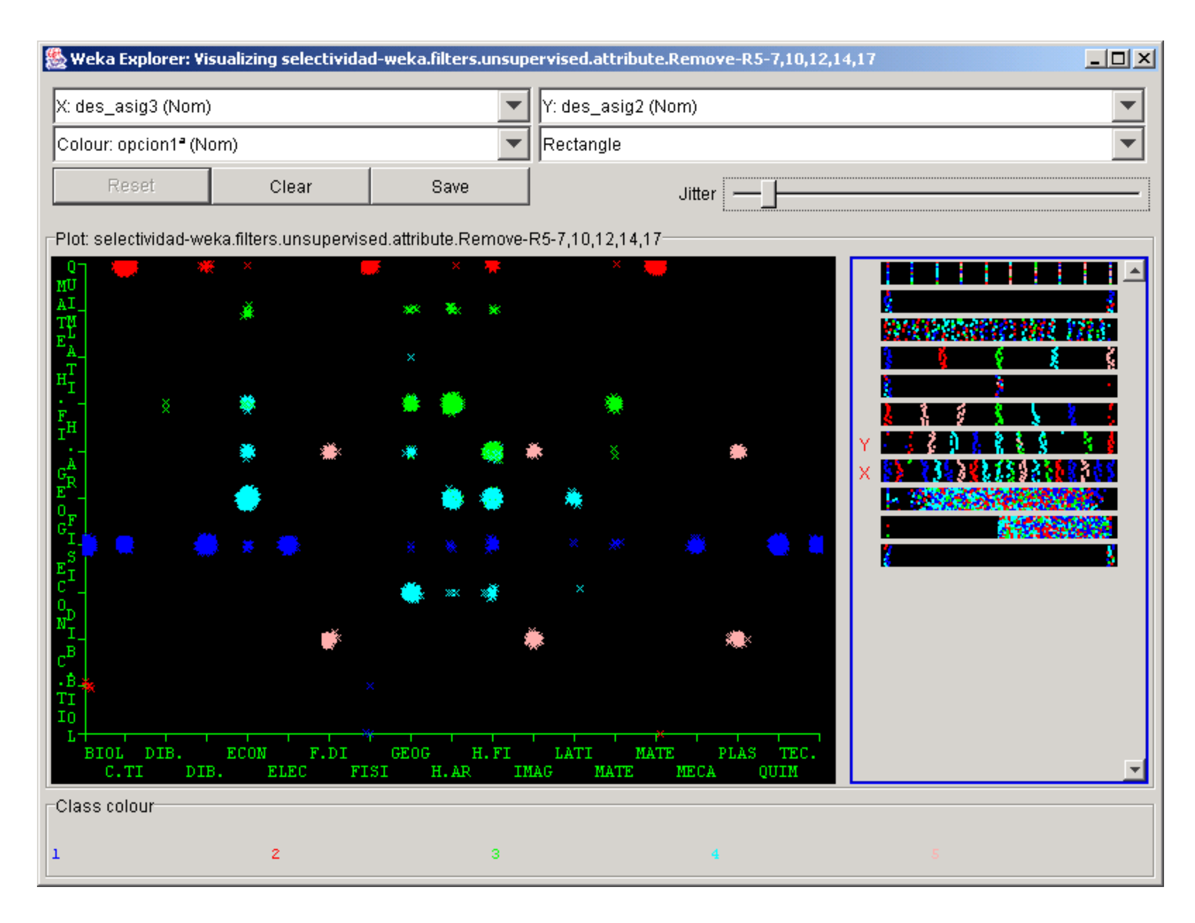


en esta gráfica podemos apreciar la relación entre ambas magnitudes, que si bien no es directa al menos define una cierta tendencia creciente, y como la convocatoria está bastante relacionada con ambas calificaciones.

Cuando lo que se relacionan son variables simbólicas, se presentan sus posibles valores a lo largo del eje. Sin embargo, en estos casos todas las instancias que comparten cada valor de un atributo simbólico pueden ocultarse

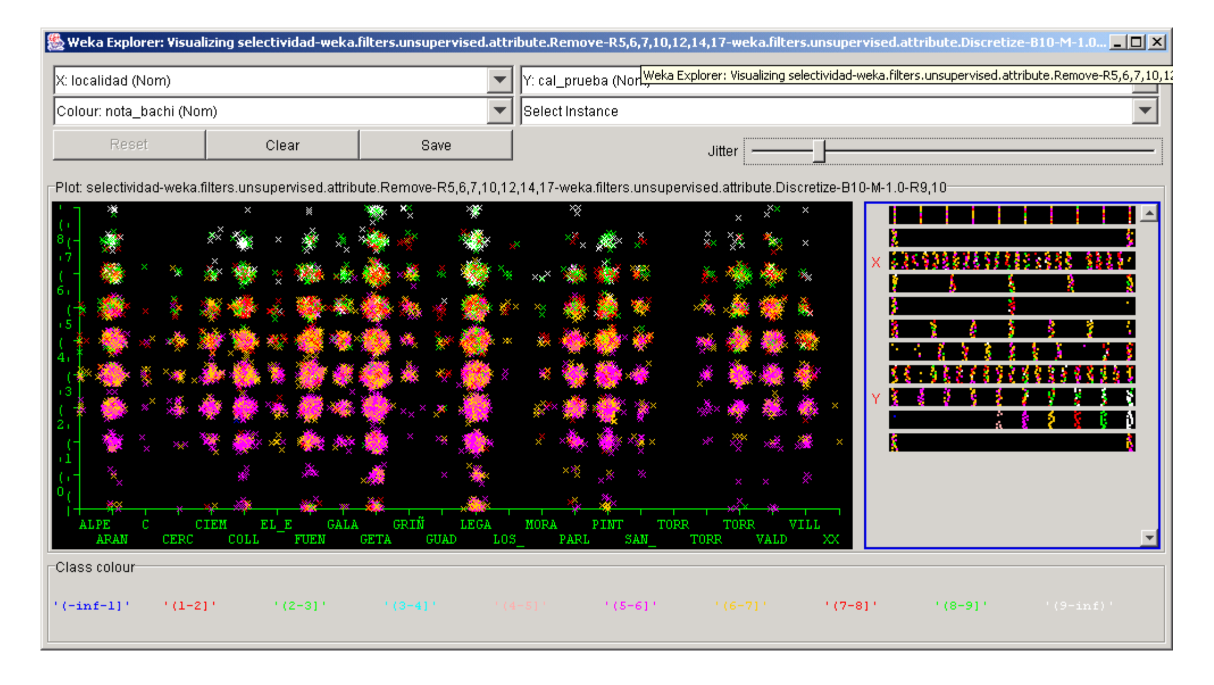
(serían un único punto en el plano), razón por la que se utiliza la facilidad de **Jitter**. Esta opción permite introducir un desplazamiento aleatorio (ruido) en las instancias, con objeto de poder visualizar todas aquellas que comparten un par de valores de atributos simbólicos, de manera que puede visualizarse la proporción de instancias que aparece en cada región. A modo de ejemplo se muestra a continuación la relación entre las tres asignaturas optativas, y con la opción cursada como color





puede verse una marcada relación entre las asignaturas opcionales, de manera que este gráfico ilustra qué tipo de asignaturas engloba cada una de las cinco posibles opciones cursadas.

Se sugiere preparar el siguiente gráfico, que relaciona la calificación obtenida en la prueba con la localidad de origen y la nota de bachillerato, estando las calificaciones discretizadas en intervalos de amplitud 2



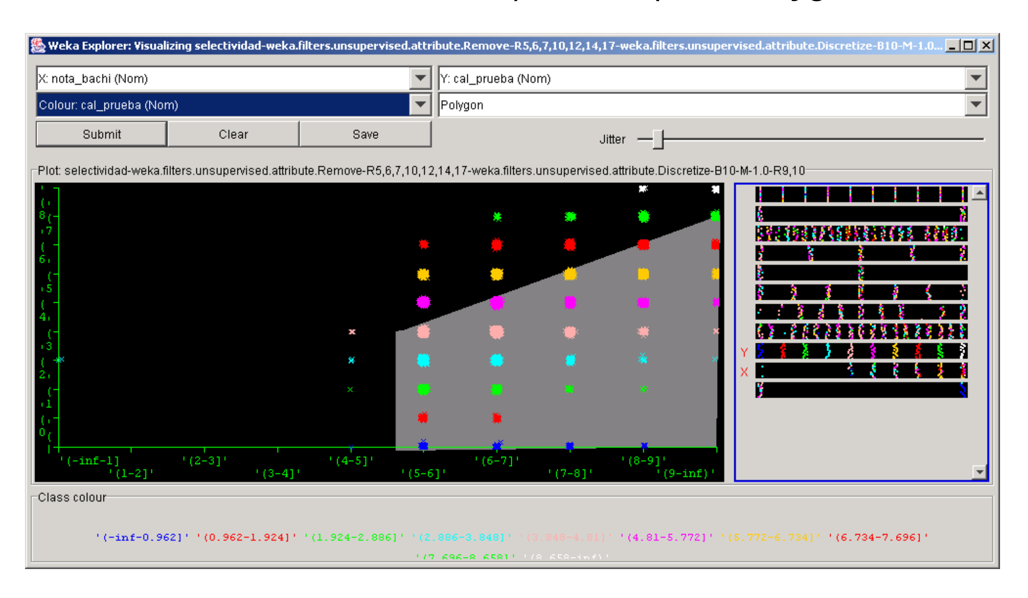
Aquí el color trae más información, pues indica en cada intervalo de calificaciones de la prueba, la calificación en bachillerato, lo que permite ilustrar la "satisfacción" con la calificación en la prueba o resultados no esperados, además distribuido por localidades.

Filtrado “gráfico” de los datos

WEKA permite también realizar filtros de selección de instancias sobre los propios gráficos, con una interacción a través del ratón para aislar los grupos de instancias cuyos atributos cumplen determinadas condiciones. Esta facilidad permite realizar filtrados de instancias de modo interactivo y más intuitivo que los filtros indicados en la sección 1.4.2.2. Las opciones que existen son:

- Selección de instancias con un valor determinado (hacer clic sobre la posición en el gráfico)
- Selección con un rectángulo de un subconjunto de combinaciones (comenzando por el vértice superior izquierdo) (**Rectangle**)
- Selección con un polígono cerrado de un subconjunto (**Polygon**)
- Selección con una línea abierta de frontera (**Polyline**)

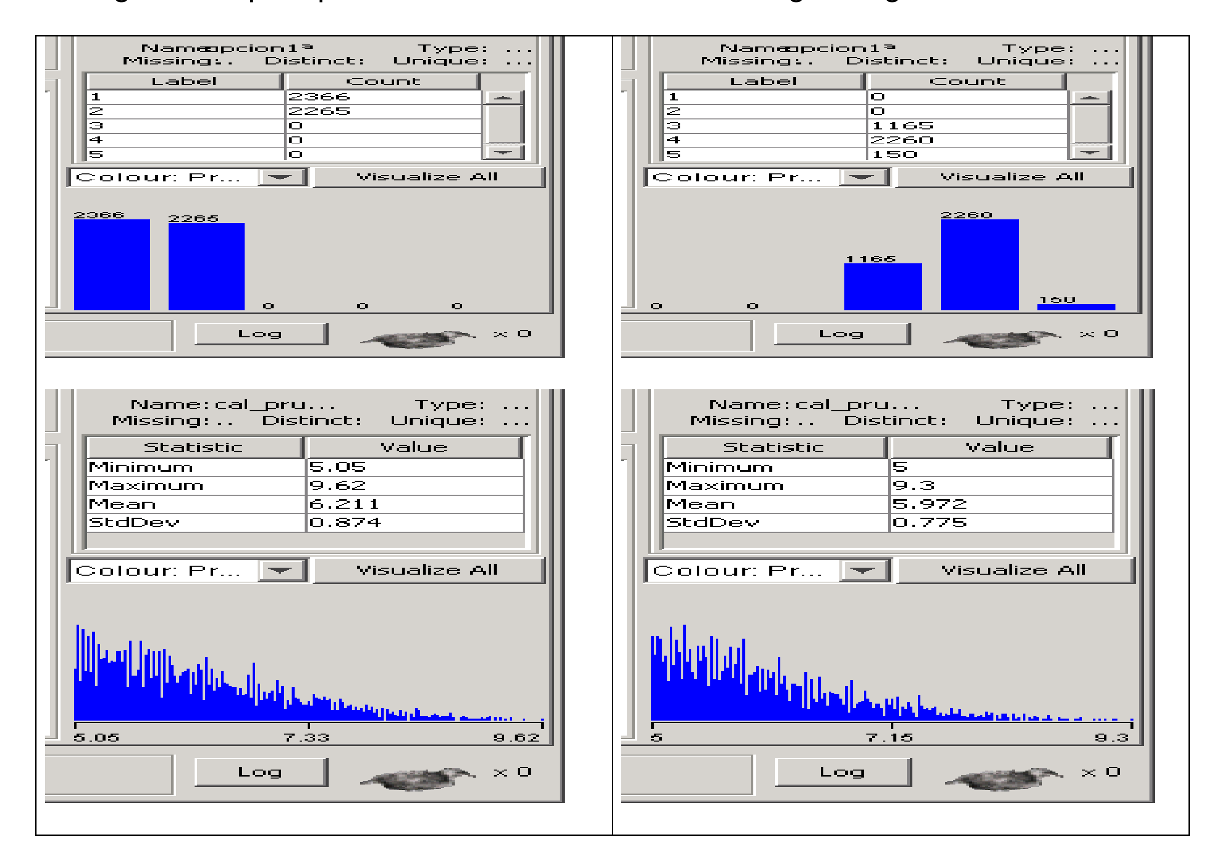
Por ejemplo, a continuación se indica la selección de alumnos que obtuvieron una calificación por debajo de sus expectativas (calificación en la prueba inferior a su nota en el bachillerato), con la opción **Polygon**.



Una vez realizada la selección, la opción **Submit** permite eliminar el resto de instancias, y **Save** almacenarlas en un fichero. **Reset** devuelve la relación a su estado original.

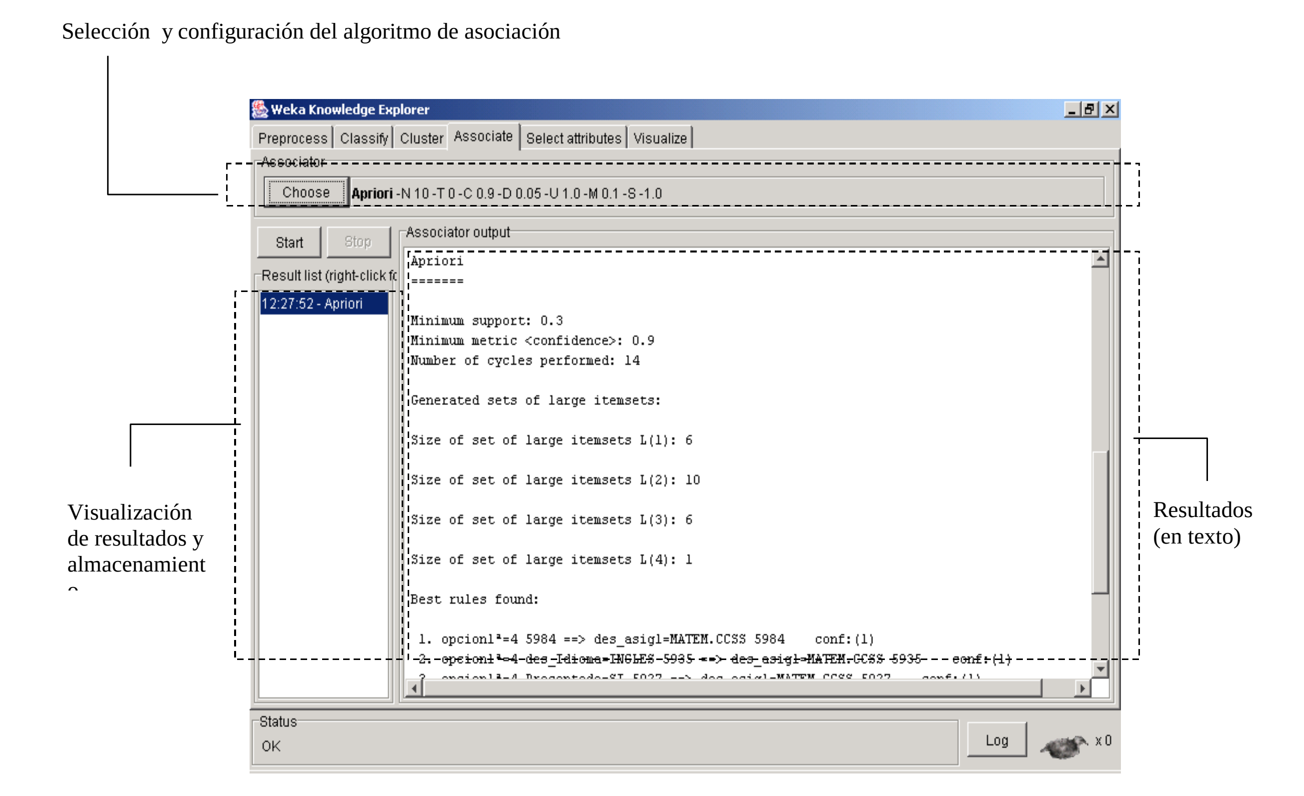
Utilice estas facilidades gráficas para hacer subconjuntos de los datos con los alumnos aprobados de las opciones 1 y 2 frente a los de las opciones 3, 4 y 5.

Salve las relaciones filtradas para a continuación cargarlas y mostrar los histogramas, que aparecerán como se indica en la figura siguiente.



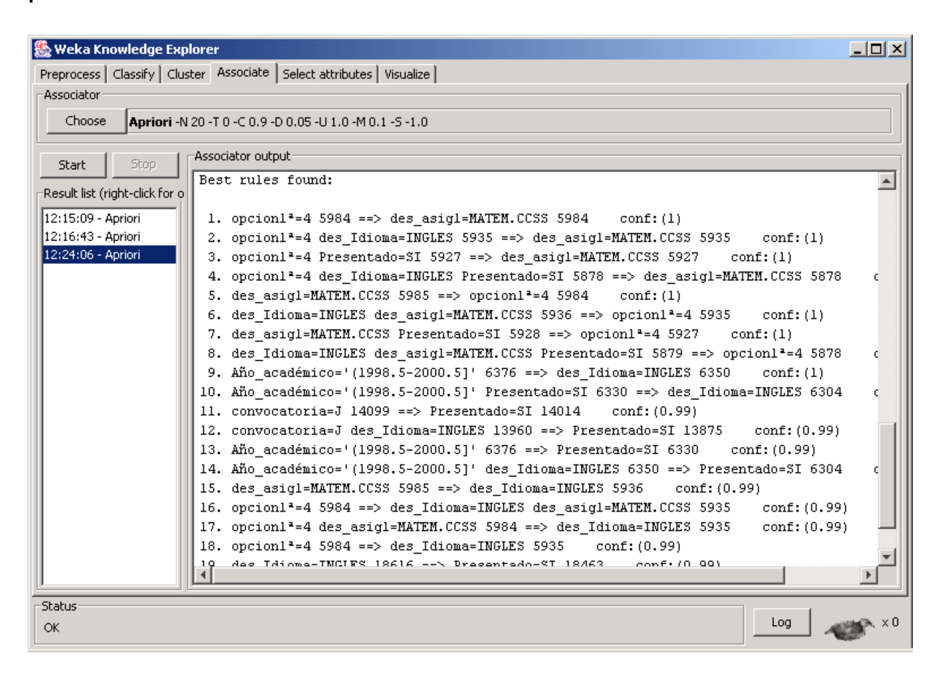
Asociación

Los algoritmos de asociación permiten la búsqueda automática de reglas que relacionan conjuntos de atributos entre sí. Son algoritmos no supervisados, en el sentido de que no existen relaciones conocidas a priori con las que contrastar la validez de los resultados, sino que se evalúa si esas reglas son estadísticamente significativas. La ventana de Asociación (**Associate** en el Explorer), tiene los siguiente elementos:

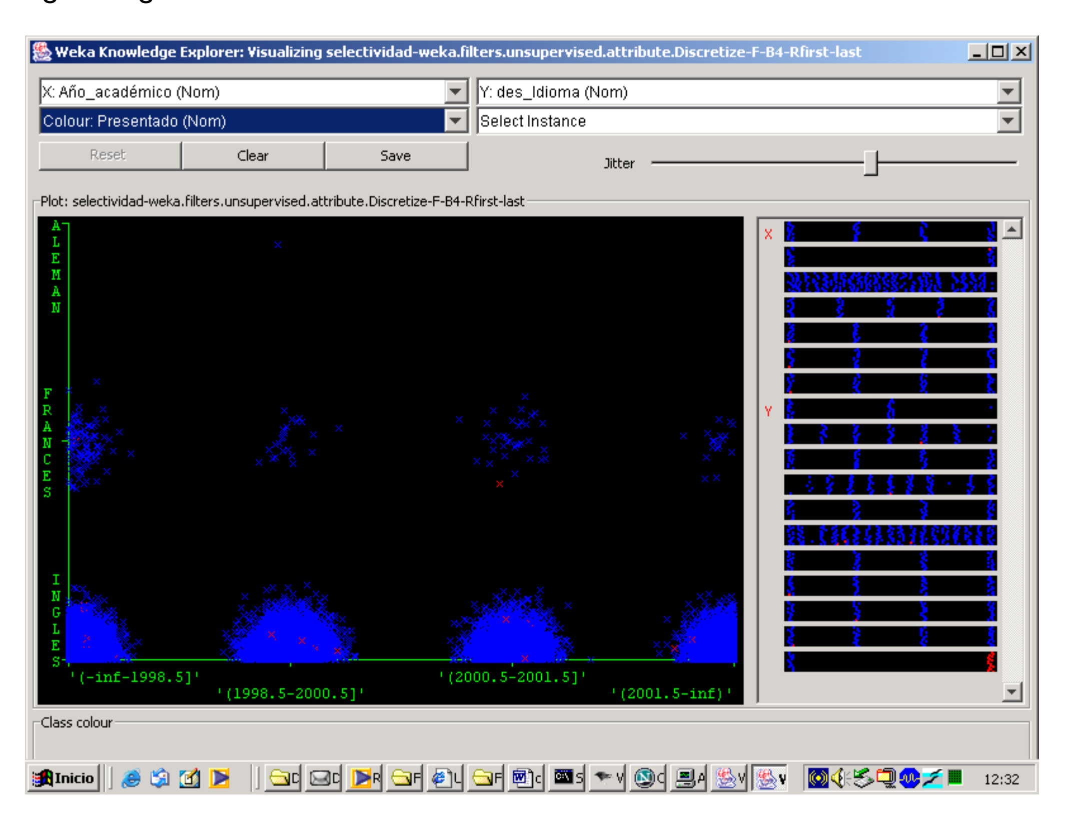


El principal algoritmo de asociación implementado en WEKA es el algoritmo "Apriori". Este algoritmo únicamente puede buscar reglas entre atributos simbólicos, razón por la que se requiere haber discretizado todos los atributos numéricos.

Por simplicidad, vamos a aplicar un filtro de discretización de todos los atributos numéricos en cuatro intervalos de la misma frecuencia para explorar las relaciones más significativas. El algoritmo lo ejecutamos con sus parámetros por defecto.



las reglas que aparecen aportan poca información. Aparecen en primer lugar las relaciones triviales entre asignaturas y opciones, así como las que relacionan suspensos en la prueba y en la calificación final. En cuanto a las que relacionan alumnos presentados con idioma seleccionado son debidas a la fuerte descompensación en el idioma seleccionado. La abrumadora mayoría de los presentados a la prueba de idioma seleccionaron el inglés, como indica la figura siguiente:



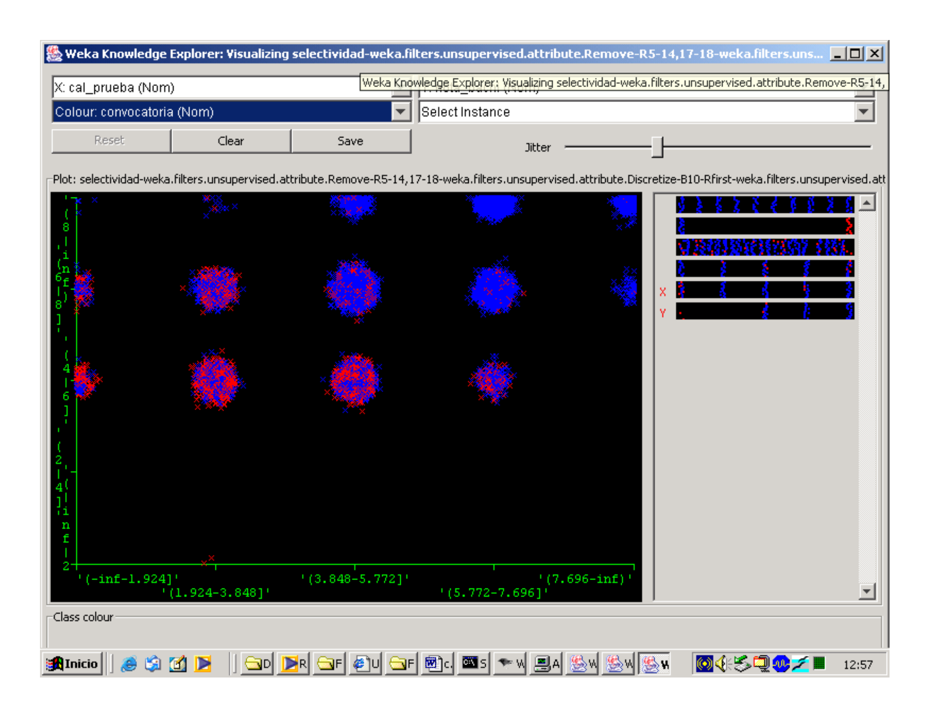
Con objeto de buscar relaciones no conocidas, se filtrarán ahora todos los atributos relacionados con descriptores de asignaturas y calificaciones parciales, quedando únicamente los atributos:

```
Año_académico
convocatoria
localidad
opcion1ª
cal_prueba
nota_bachi
```

En este caso, las reglas más significativas son:

```
1. nota_bachi='(8-inf)' 2129 ==> convocatoria=J 2105    conf:(0.99)
2. cal_prueba='(5.772-7.696]' nota_bachi='(6-8]' 2521 ==>
   convocatoria=J 2402    conf:(0.95)
3. cal_prueba='(5.772-7.696]' 4216 ==>
   convocatoria=J 3997    conf:(0.95)
```

estas reglas aportan información no tan trivial: el 99% de alumnos con nota superior a 8 se presentan a la convocatoria de Junio, así el 95% de los alumnos con calificación en la prueba entre 5.772 y 7.



es significativo ver que no aparece ninguna relación importante entre las calificaciones, localidad y año de la convocatoria. También es destacado ver la ausencia de efecto de la opción cursada.

Si preparamos los datos para dejar sólo cinco atributos,

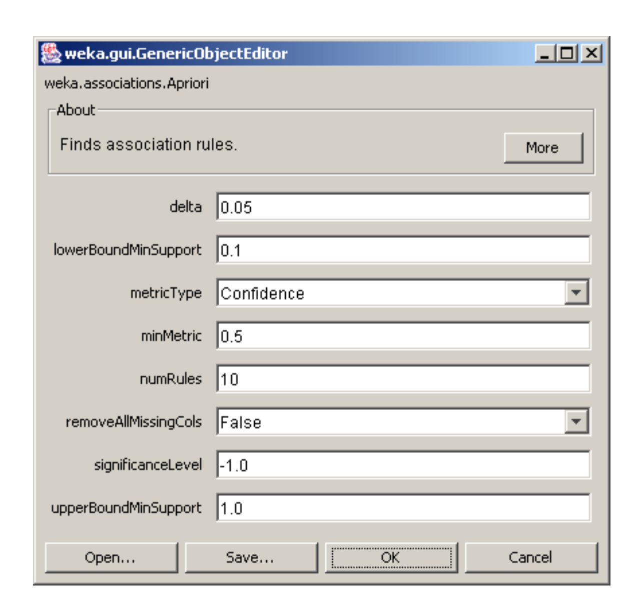
```
Año_académico
convocatoria
localidad
opcion1ª
cal_final,
```

con el último discretizado en dos grupos iguales (hasta 5.85 y 5.85 hasta 10), tenemos que de nuevo las reglas más significativas relacionan convocatoria con calificación, pero ahora entran en juego opciones y localidades, si bien bajando la precisión de las reglas:

```
1. opcion1ª=1 cal_final='(5.685-inf)' 2810 ==>
    convocatoria=J 2615    conf:(0.93)
2. localidad=LEGANES cal_final='(5.685-inf)' 2514 ==>
    convocatoria=J 2315    conf:(0.92)
3. Año_académico='(1998.4-2000.2]' cal_final='(5.685-inf)' 3175 ==>
    convocatoria=J 2890    conf:(0.91)
4. cal_final='(5.685-inf)' 9397 ==>
    convocatoria=J 8549    conf:(0.91)
5. opcion1ª=4 cal_final='(5.685-inf)' 2594 ==>
    convocatoria=J 2358    conf:(0.91)
6. Año_académico='(2000.2-inf)' cal_final='(5.685-inf)' 3726 ==>
```

```
   convocatoria=J 3376    conf:(0.91)
7. localidad=GETAFE cal_final='(5.685-inf)' 2156 ==>
   convocatoria=J 1951    conf:(0.9)
```

Al filtrar la convocatoria, que nos origina relaciones bastante evidentes, tendremos las reglas más significativas entre localidad, año, calificación y opción. Como podemos ver, al lanzar el algoritmo con los parámetros por defecto no aparece ninguna regla. Esto es debido a que se forzó como umbral mínimo aceptable para una regla el 90%. Vamos a bajar ahora este parámetro hasta el 50%:




```
Best rules found:

 1. opcion1ª=4 5984 ==> cal_final='(-inf-5.685]' 3390    conf:(0.57)
 2. opcion1ª=1 5131 ==> cal_final='(5.685-inf)' 2810    conf:(0.55)
 3. Año_académico='(2000.2-inf)' 7049 ==>
    cal_final='(5.685-inf)' 3726    conf:(0.53)
 4. opcion1ª=2 4877 ==> cal_final='(5.685-inf)' 2575    conf:(0.53)
 5. localidad=GETAFE 4464 ==>
    cal_final='(-inf-5.685]' 2308    conf:(0.52)
 6. localidad=LEGANES 4926 ==>
   cal_final='(5.685-inf)' 2514    conf:(0.51)
 7. Año_académico='(1998.4-2000.2]' 6376 ==>
   cal_final='(-inf-5.685]' 3201    conf:(0.5)
```

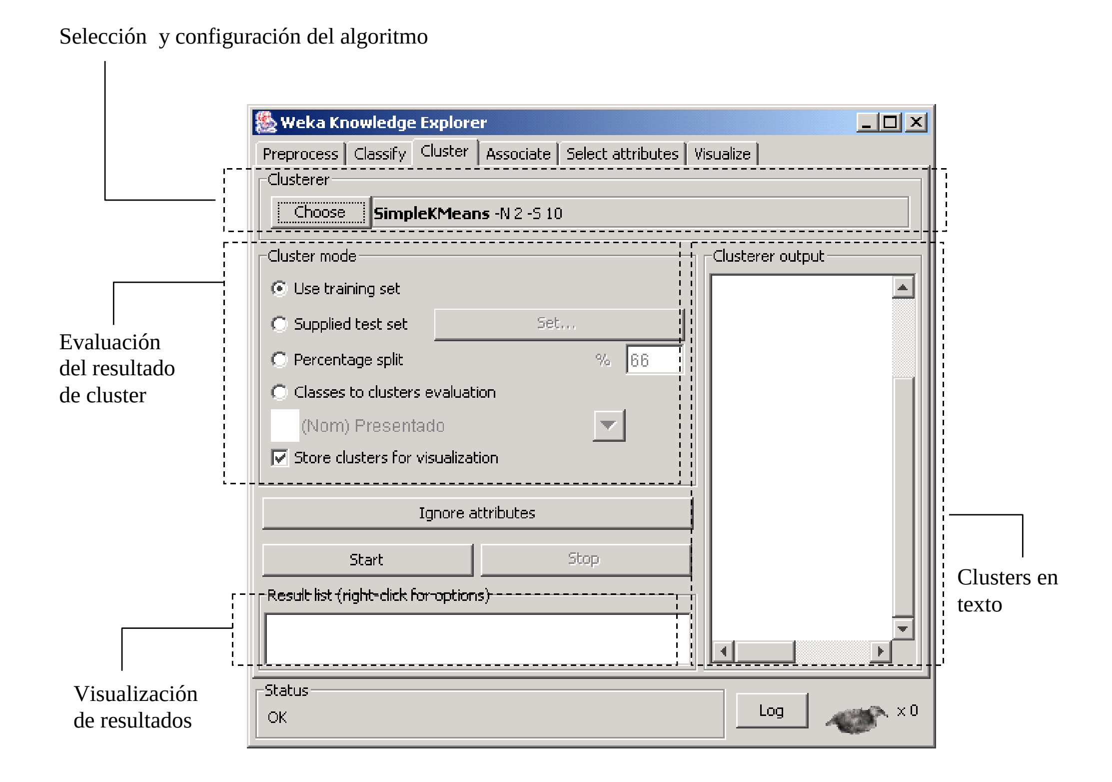
Por tanto, forzando los términos, tenemos que los estudiantes de las 2 primeras opciones tienen mayor probabilidad de aprobar la prueba, así como los estudiantes de la localidad de Leganés. Los estudiantes de Getafe tienen una probabilidad superior de obtener una calificación inferior. Hay que destacar que estas reglas rozan el umbral del 50%, pero han sido seleccionadas como las más significativas de todas las posibles. También hay que considerar que si aparecen estas dos localidades en primer lugar es simplemente por su mayor volumen de datos, lo que otorga una significatividad superior en las relaciones encontradas. Si se consulta la bibliografía, el primer criterio de selección de reglas del algoritmo "A priori" es la precisión o confianza, dada por el porcentaje de veces que instancias que cumplen el antecedente cumplen el consecuente, pero el segundo es el soporte, dado por el número de instancias

sobre las que es aplicable la regla. En todo caso, son reglas de muy baja precisión y que habría que considerar simplemente como ciertas tendencias.

Agrupamiento

La opción **Cluster** del *Experimenter* nos permite aplicar algoritmos de agrupamiento de instancias a nuestros datos. Estos algoritmos buscan grupos de instancias con características "similares", según un criterio de comparación entre valores de atributos de las instancias definidos en los algoritmos.

El mecanismo de selección, configuración y ejecución es similar a otros elementos: primero se selecciona el algoritmo con **Choose**, se ajustan sus parámetros seleccionando sobre el área donde aparece, y se después se ejecuta. El área de agrupamiento del Explorer presenta los siguientes elementos de configuración:



Una vez que se ha realizado la selección y configuración del algoritmo, se puede pulsar el botón **Start**, que hará que se aplique sobre la relación de trabajo. Los resultados se presentarán en la ventana de texto de la parte derecha. Además, la ventana izquierda permite listar todos los algoritmos y resultados que se hayan ejecutado en la sesión actual. Al seleccionarlos en esta lista de visualización se presentan en la ventana de texto a la derecha, y además se permite abrir ventanas gráficas de visualización con un menú contextual que aparece al pulsar el botón derecho sobre el resultado seleccionado. Por último, en esta opción de Agrupamiento aparecen las siguientes opciones adicionales en la pantalla.

Ignorar atributos

La opción **Ignoring Attributes** permite sacar fuera atributos que no interesa considerar para el agrupamiento, de manera que el análisis de parecido entre instancias no considera los atributos seleccionados. Al accionar esta opción aparecen todos los atributos disponibles. Se pueden seleccionar con el botón izquierdo sobre un atributo específico, o seleccionar grupos usando SHIFT para un grupo de atributos contiguos y CONTROL para grupos de atributos sueltos.

Evaluación

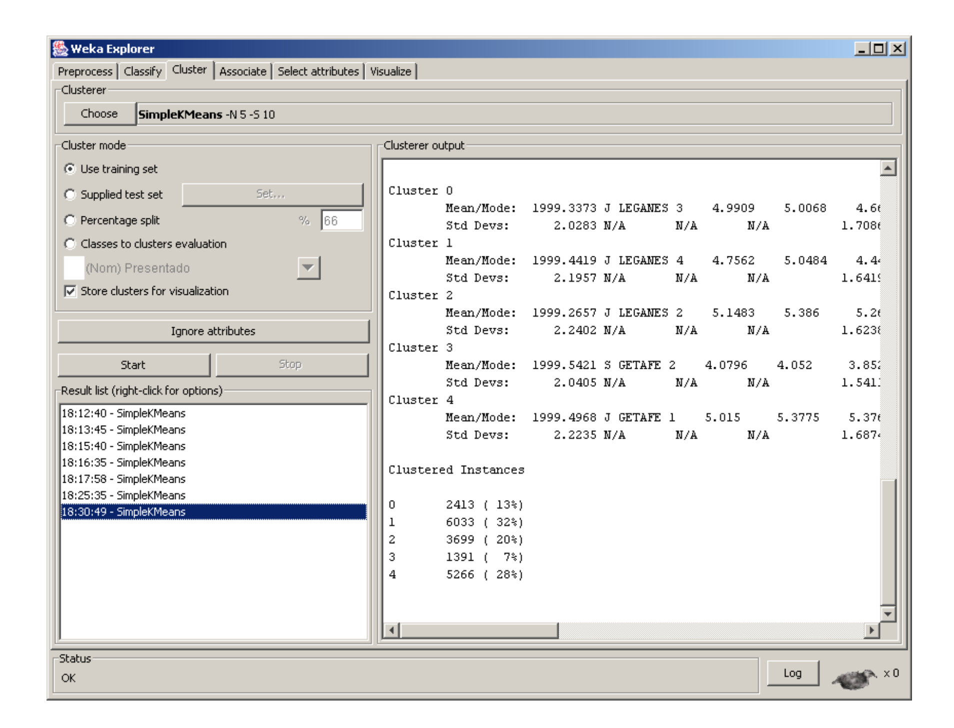
La opción **Cluster Mode** permite elegir como evaluar los resultados del agrupamiento. Lo más simple es utilizar el propio conjunto de entrenamiento, **Use tranining set**, que indica que porcentaje de instancias se van a cada grupo. El resto de opciones realizan un entrenamiento con un conjunto, sobre el que construyen los clusters y a continuación aplican estos clusters para clasificar un conjunto independiente que puede proporcionarse aparte (**Supplied test**), o ser un porcentaje del conjunto de entrada (**Percentage split**). Existe también la opción de comparar los clusters con un atributo de clasificación (**Classes to clusters evaluation**) que no se considera en la construicción de los clusters. Nosotros nos centraremos únicamente en la primera opción, dejando el resto de opciones de evaluación para más adelante, cuando lleguemos a los algoritmos de clasificación.

Finalmente, el cuadro opcional de almacenamiento de instancias, **Store clusters for visualization**, es muy útil para después analizar los resultados gráficamente.

Agrupamiento numérico

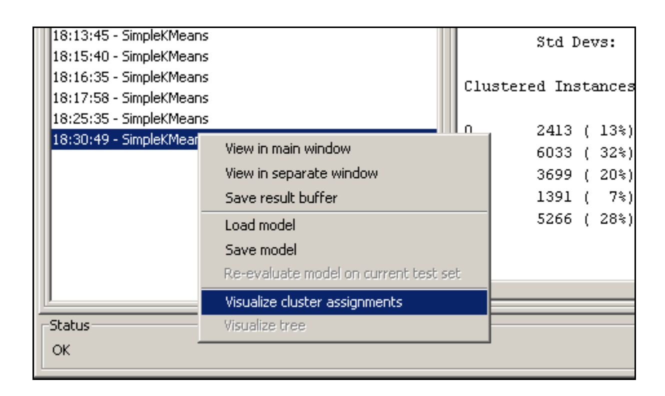
En primer lugar utilizaremos el algoritmo de agrupamiento K-medias, por ser uno de los más veloces y eficientes, si bien uno de los más limitados. Este algoritmo precisa únicamente del número de categorías similares en las que queremos dividir el conjunto de datos. Suele ser de interés repetir la ejecución del algoritmo K-medias con diferentes semillas de inicialización, dada la notable dependencia del arranque cuando no está clara la solución que mejor divide el conjunto de instancias.

En nuestro ejemplo, vamos a comprobar si el atributo “opción” divide naturalmente a los alumnos en grupos similares, para lo que seleccionamos el algoritmo **SimpleKMeans** con parámetro **numClusters** con valor 5. Los resultados aparecen en la ventana de texto derecha:

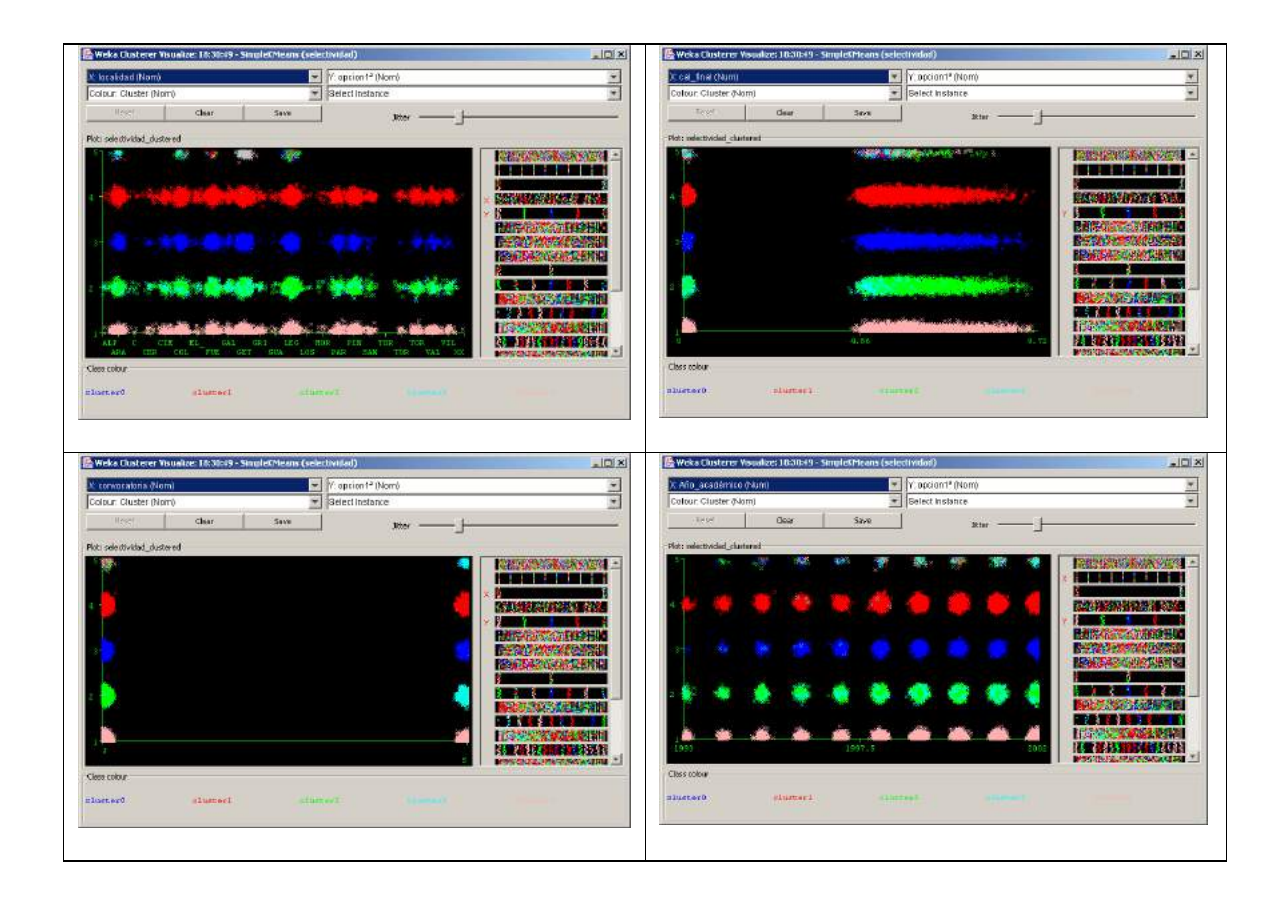


Nos aparecen los 5 grupos de ejemplos más similares, y sus centroides (promedios para atributos numéricos, y valores más repetidos en cada grupo para atributos simbólicos).

En este caso es de interés analizar gráficamente como se distribuyen diferentes valores de los atributos en los grupos generados. Para ello basta pulsar con botón derecho del ratón sobre el cuadro de resultados, y seleccionar la opción **visualizeClusterAssignments**

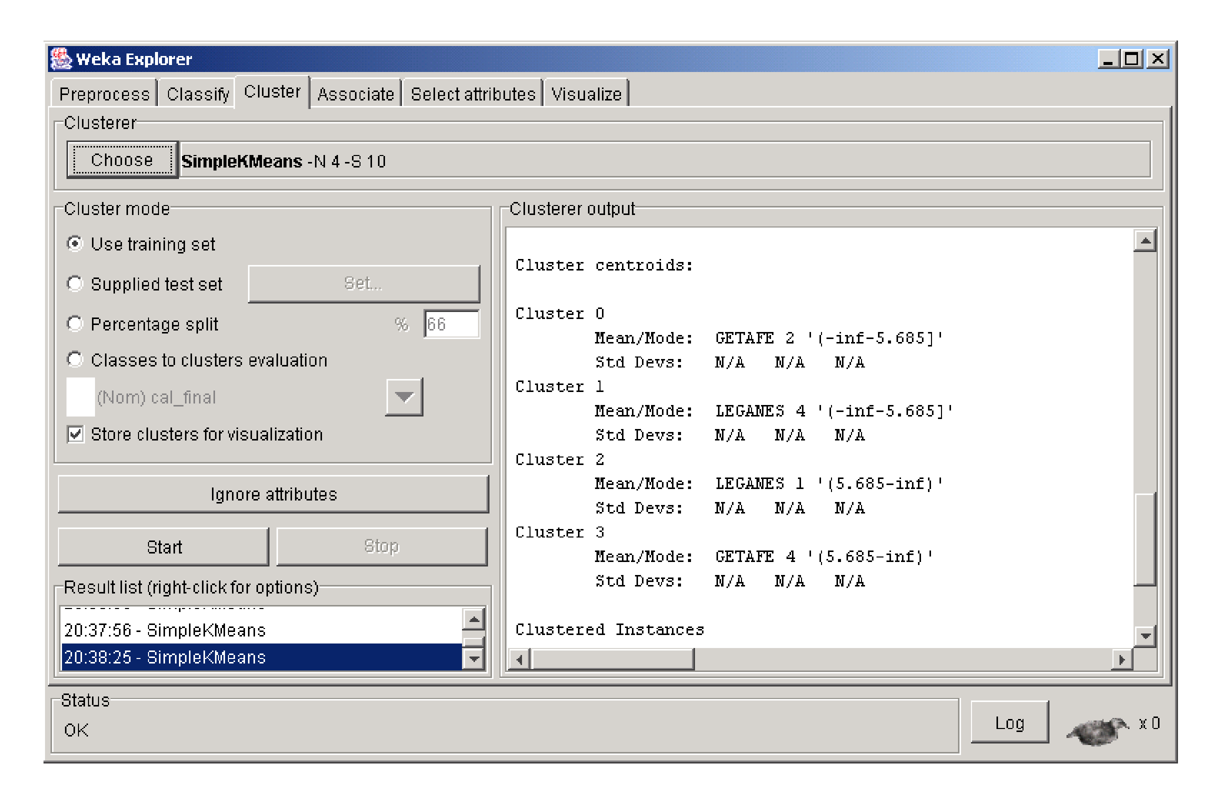


Si seleccionamos combinaciones del atributo opción con localidad, nota o convocatoria podemos ver la distribución de grupos:

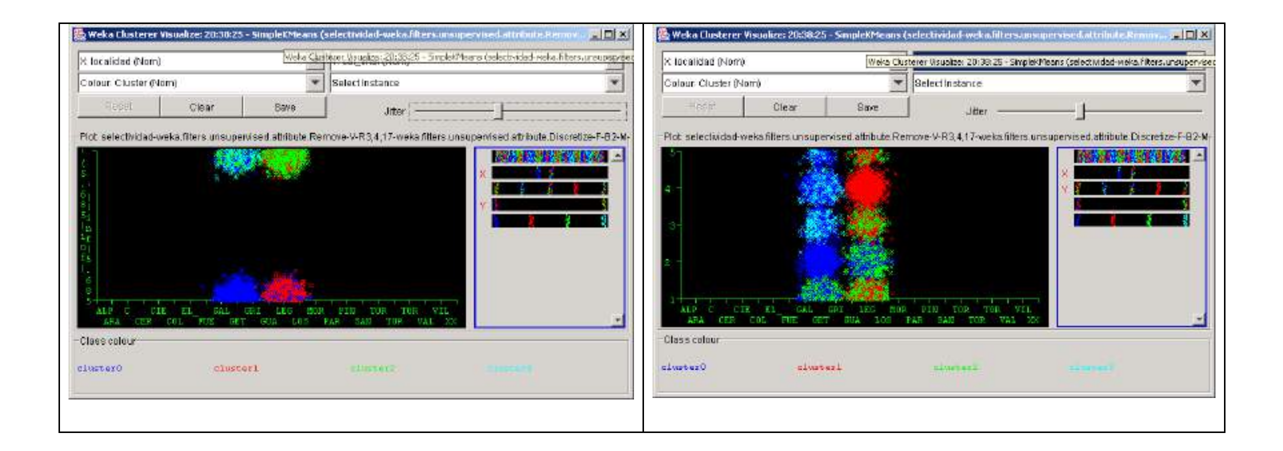


A la vista de estos gráficos podemos concluir que el “parecido” entre casos viene dado fundamentalmente por las opciones seleccionadas. Los clusters 0, 1 y 4 se corresponden con las opciones 3, 4 y 1, mientras que los clusters 2 y 3 representan la opción 3 en las convocatorias de junio y septiembre.

Aprovechando esta posibilidad de buscar grupos de semejanzas, podríamos hacer un análisis más particularizado a las dos localidades mayores, Leganés y Getafe, buscando qué opciones y calificaciones aparecen con más frecuencia. Vamos a preparar los datos con filtros de modo que tengamos únicamente tres atributos: localidad, opción, y calificación final. Además, discretizamos las calificaciones en dos grupos de la misma frecuencia (estudiantes con mayor y menor éxito), y únicamente nos quedamos con los alumnos de Leganés y Getafe. Utilizaremos para ello los filtros adecuados. A continuación aplicamos el algoritmo K-medias con 4 grupos.



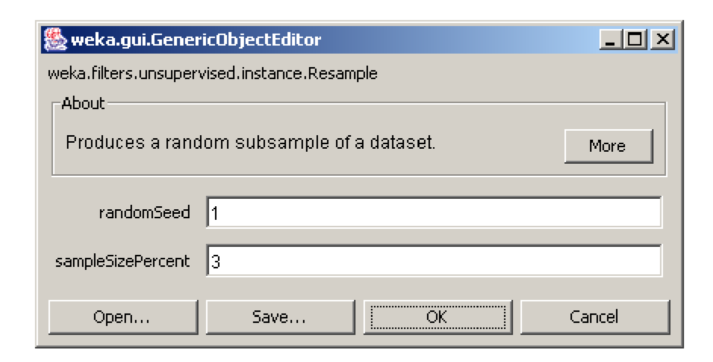
vemos que los grupos nos muestran la presencia de buenos alumnos en Getafe en la opción 4, y buenos alumnos en Leganés en la opción 1, siempre considerando estas conclusiones como tendencias en promedio. Gráficamente vemos la distribución de clusters por combinaciones de atributos:



Si consideramos que en Leganés hay escuelas de ingeniería, y en Getafe facultades de Humanidades, podríamos concluir que podría ser achacable al impacto de la universidad en la zona.

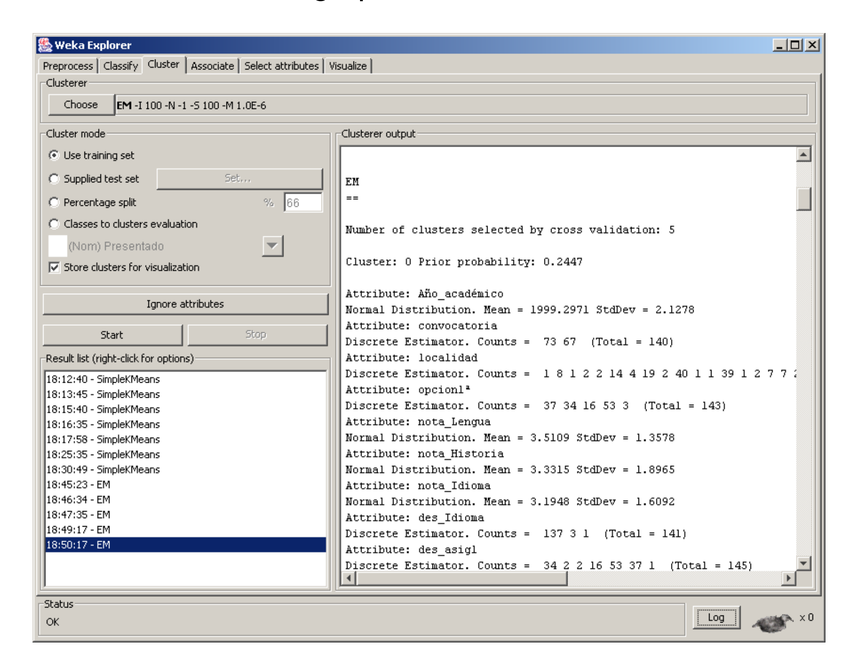
El algoritmo EM proviene de la estadística y es bastante más elaborado que el K-medias, con el coste de que requiere muchas más operaciones, y es apropiado cuando sabemos que los datos tienen una variabilidad estadística de modelo conocido. Dada esta mayor complejidad, y el notable volumen del

fichero de datos, primero aplicaremos un filtro de instancias al 3% para dejar un número de 500 instancias aproximadamente. Para esto último iremos al preprocesado y aplicamos un filtro de instancias, el filtro **Resample**, con factor de reducción al 3%:

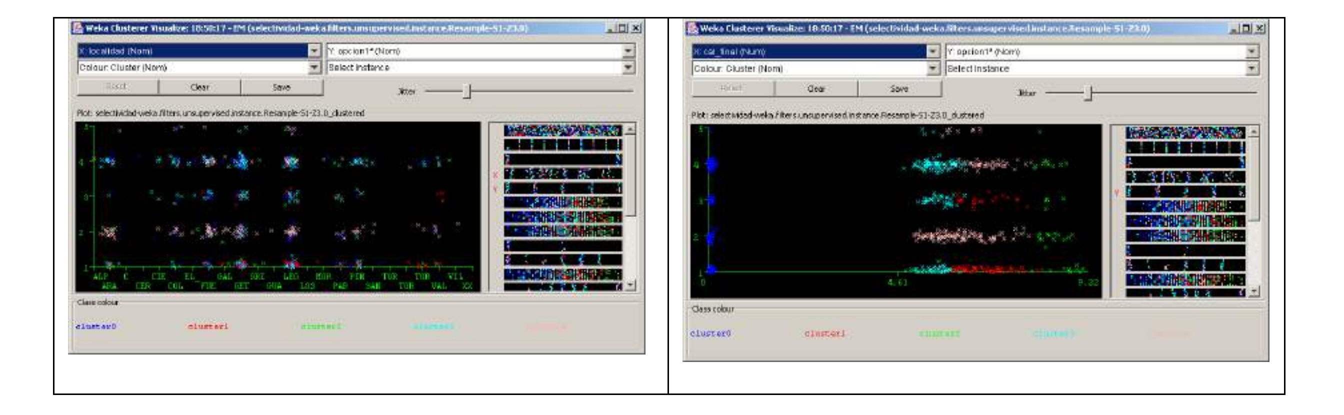


Una ventaja adicional del algoritmo de clustering EM es que permite además buscar el número de grupos más apropiado, para lo cual basta indicar a –1 el número de clusters a formar, que es la opción que viene por defecto. Esto se interpreta como dejar el parámetro del número de clusters como un valor a optimizar por el propio algoritmo.

Tras su aplicación, este algoritmo determina que hay cinco clusters significativos en la muestra de 500 alumnos, y a continuación indica los centroides de cada grupo:



Al igual que antes, es interesante analizar el resultado del agrupamiento sobre diferentes combinaciones de atributos, haciendo uso de la facilidad **visualizeClusterAssignments**



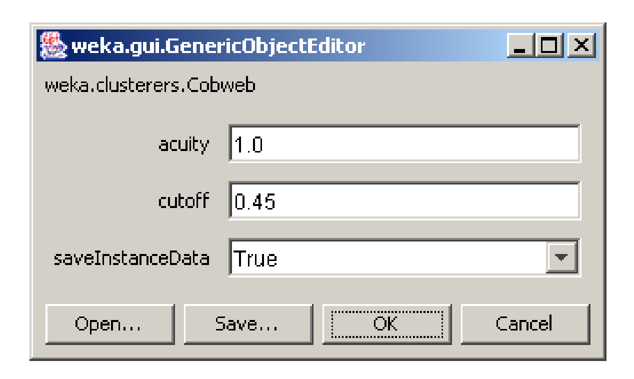
Por tanto podría concluirse que para este segundo algoritmo de agrupamiento por criterios estadísticos y no de distancias entre vectores de atributos, predomina el agrupamiento de los alumnos básicamente por tramos de calificaciones, independientemente de la opción, mientras que en el anterior pesaba más el perfil de asignaturas cursado que las calificaciones.

Esta disparidad sirve para ilustrar la problemática de la decisión del criterio de "parecido" entre instancias para realizar el agrupamiento.

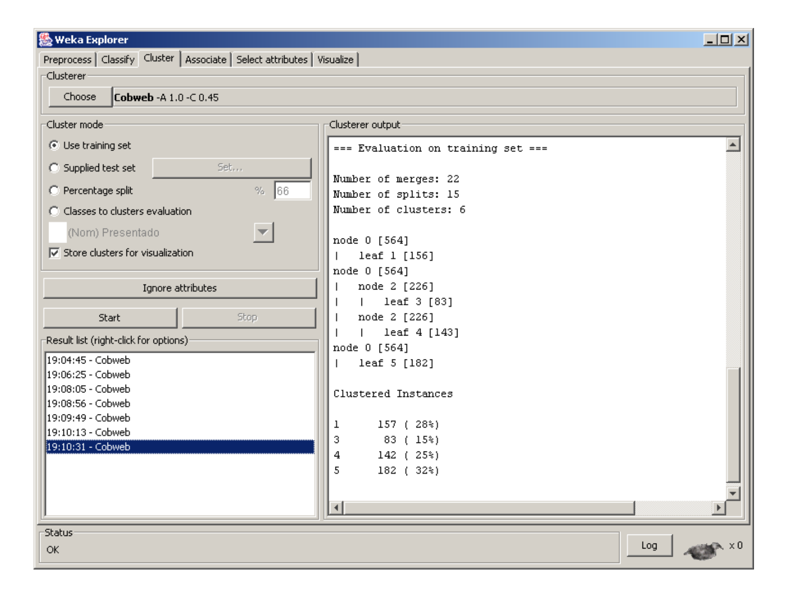
Agrupamiento simbólico

Finalmente, como alternativa a los algoritmos de agrupamiento anteriores, el agrupamiento simbólico tiene la ventaja de efectuar un análisis cualitativo que construye categorías jerárquicas para organizar los datos. Estas categorías se forman con un criterio probabilístico de "utilidad", llegando a las que permiten homogeneidad de los valores de los atributos dentro de cada una y al mismo tiempo una separación entre categorías dadas por los atributos, propagándose estas características en un árbol de conceptos.

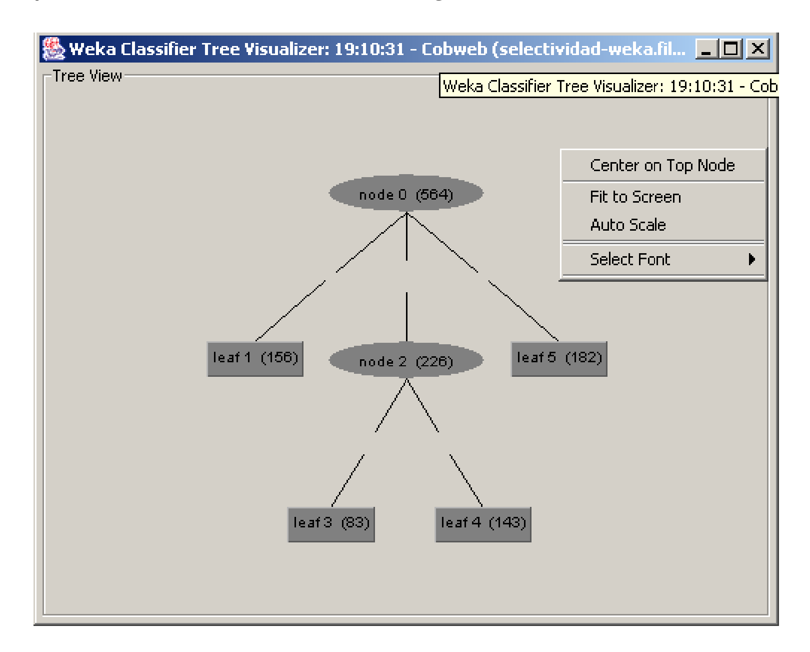
Si aplicamos el algoritmo cobweb con los parámetros por defecto sobre la muestra reducida de instancias (dada la complejidad del algoritmo), el árbol generado llega hasta 800 nodos. Vamos a modificar el parámetro **cut-off**, que permite poner condiciones más restrictivas a la creación de nuevas categorías en un nivel y subcategorías. Con los parámetros siguientes se llega a un árbol muy manejable:



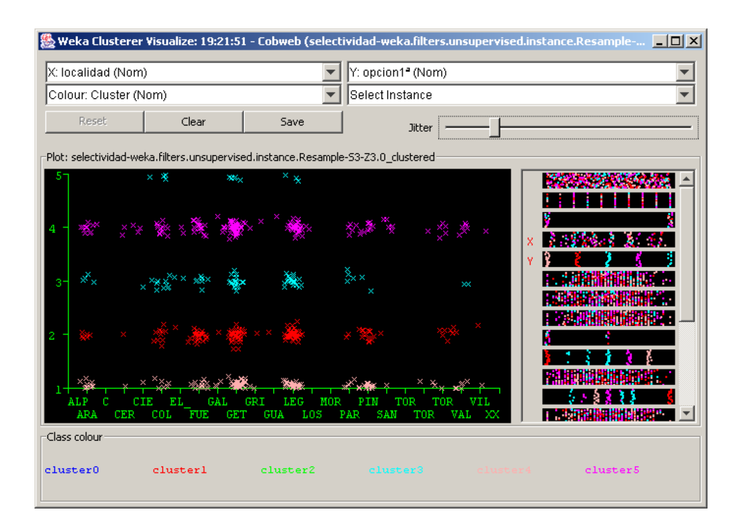
la opción **saveInstanceData** es de gran utilidad para después analizar la distribución de valores de atributos entre las instancias de cada parte del árbol de agrupamiento. Una vez ejecutado Cobweb, genera un resultado como el siguiente:



hay 3 grupos en un primer nivel, y el segundo se subdivide en otros dos. De nuevo activando el botón derecho sobre la ventana de resultados, ahora podemos visualizar el árbol gráficamente:



las opciones de visualización aparecen al pulsar el botón derecho en el fondo de la figura. Se pueden visualizar las instancias que van a cada nodo sin más que pulsar el botón derecho sobre él. Si nos fijamos en como quedan distribuidas las instancias por clusters, con la opción **visualizeClusterAssignments,** llegamos a la figura:

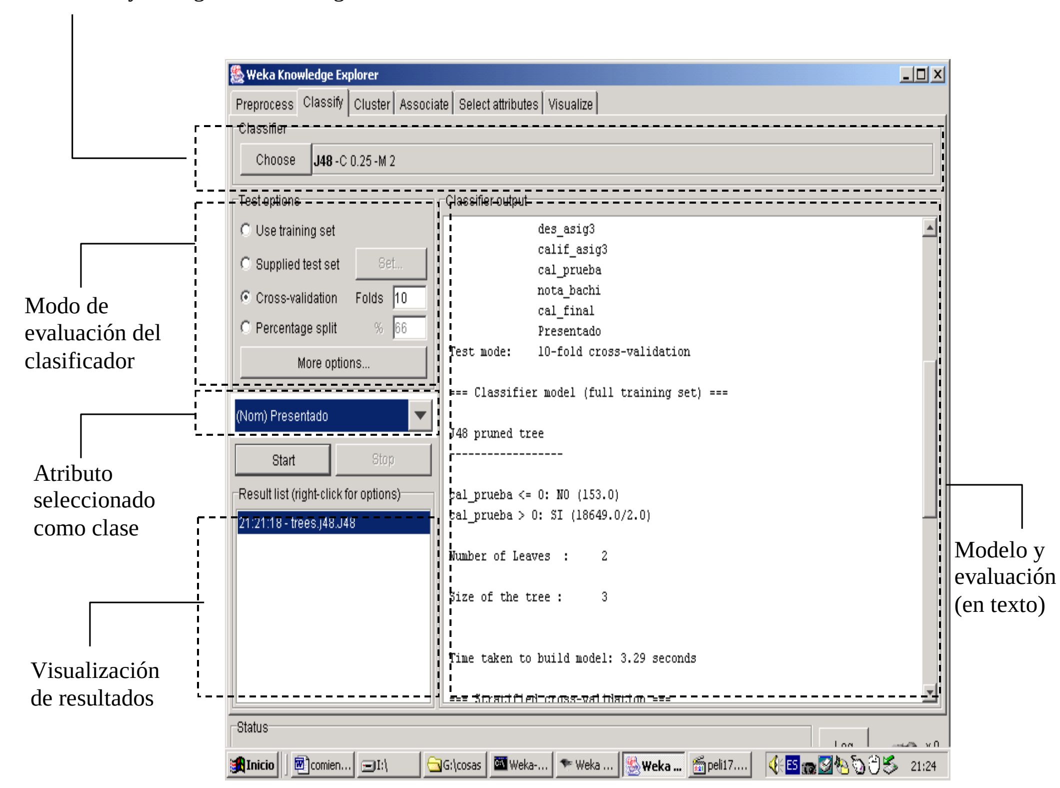


por tanto, vemos que de nuevo vuelve a pesar la opción como criterio de agrupamiento. Los nodos hoja 1, 3, 4 y 5 se corresponden con las opciones cursadas 2, 3, 1 y 4 respectivamente. En un primer nivel hay tres grupos, uno para la opción 2, otro para la opción 4 y otro que une las opciones 1 y 3. Este último se subdivide en dos grupos que se corresponden con ambas opciones.

Clasificación

Finalmente, en esta sección abordamos el problema de la clasificación, que es el más frecuente en la práctica. En ocasiones, el problema de clasificación se formula como un refinamiento en el análisis, una vez que se han aplicado algoritmos no supervisados de agrupamiento y asociación para describir relaciones de interés en los datos.

Se pretende construir un modelo que permita predecir la categoría de las instancias en función de una serie de atributos de entrada. En el caso de WEKA, la clase es simplemente uno de los atributos simbólicos disponibles, que se convierte en la variable objetivo a predecir. Por defecto, es el último atributo (última columna) a no ser que se indique otro explícitamente. La configuración de la clasificación se efectúa con la ventana siguiente:



la parte superior, como es habitual sirve para seleccionar el algoritmo de clasificación y configurarlo. El resto de elementos a definir en esta ventana se describen a continuación.

Modos de evaluación del clasificador

El resultado de aplicar el algoritmo de clasificación se efectúa comparando la clase predicha con la clase real de las instancias. Esta evaluación puede realizarse de diferentes modos, según la selección en el cuadro **Test options**:

- **Use training set:** esta opción evalúa el clasificador sobre el mismo conjunto sobre el que se construye el modelo predictivo para determinar el error, que en este caso se denomina "error de resustitución". Por tanto, esta opción puede proporcionar una estimación demasiado optimista del comportamiento del clasificador, al evaluarlo sobre el mismo conjunto sobre el que se hizo el modelo.

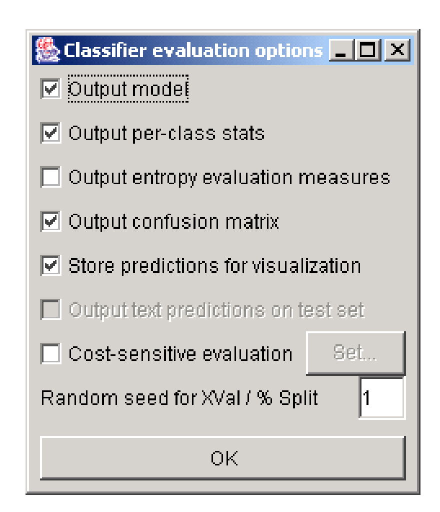
- **Supplied test set**: evaluación sobre conjunto independiente. Esta opción permite cargar un conjunto nuevo de datos. Sobre cada dato se realizará una predicción de clase para contar los errores.

- **Cross-validation**: evaluación con validación cruzada. Esta opción es la más elaborada y costosa. Se realizan tantas evaluaciones como se indica en el parámetro **Folds**. Se dividen las instancias en tantas carpetas como indica este parámetro y en cada evaluación se toman las instancias de cada carpeta como datos de test, y el resto como datos de entrenamiento para

construir el modelo. Los errores calculados son el promedio de todas las ejecuciones.

- **Percentage split :** esta opción divide los datos en dos grupos, de acuerdo con el porcentaje indicado (**%**). El valor indicado es el porcentaje de instancias para construir el modelo, que a continuación es evaluado sobre las que se han dejado aparte. Cuando el número de instancias es suficientemente elevado, esta opción es suficiente para estimar con precisión las prestaciones del clasificador en el dominio.

Además de estas opciones para seleccionar el modo de evaluación, el botón **More Options** abre un cuadro con otras opciones adicionales:



Output model: permite visualizar (en modo texto y, con algunos algoritmos, en modo gráfico) el modelo construido por el clasificador (árbol, reglas, etc.)

Output per-class stats: obtiene estadísticas de los errores de clasificación por cada uno de los valores que toma el atributo de clase

Output entropy evaluation measures: generaría también medidas de evaluación de entropía

Store predictions for visualization: permite analizar los errores de clasificación en una ventana de visualización

Cost-sensitive evaluation: con esta opción se puede especificar una función con costes relativos de los diferentes errores, que se rellena con el botón **Set**

en nuestro ejemplo utilizaremos los valores por defecto de estas últimas opciones.

Evaluación del clasificador en ventana de texto

Una vez se ejecuta el clasificador seleccionado sobre los datos de la relación, en la ventana de texto de la derecha aparece información de ejecución, el modelo generado con todos los datos de entrenamiento y los resultados de la

evaluación. Por ejemplo, al predecir el atributo "presentado", con un árbol de decisión de tipo J48, aparece el modelo textual siguiente:

```
J48 pruned tree
------------------

cal_prueba <= 0: NO (153.0)
cal_prueba > 0: SI (18649.0/2.0)

Number of Leaves  :     2

Size of the tree :      3
```

Se obtiene a partir de los datos esta relación trivial, salvo dos únicos casos de error: los presentados son los que tienen una calificación superior a 0. Con referencia al informe de evaluación del clasificador, podemos destacar tres elementos:

- **Resumen** (*Summary*): es el porcentaje global de errores cometidos en la evaluación

- **Precisión detallada por clase**: para cada uno de los valores que puede tomar el atributo de clase: el porcentaje de instancias con ese valor que son correctamente predichas (TP: true positives), y el porcentaje de instancias con otros valores que son incorrectamente predichas a ese valor aunque tenían otro (FP: false positives). Las otras columnas, *precision, recall, F-measure*, se relacionan con estas dos anteriores.

- **Matriz de confusión**: aquí aparece la información detallada de cuantas instancias de cada clase son predichas a cada uno de los valores posibles. Por tanto, es una matriz con N^2 posiciones, con N el número de valores que puede tomar la clase. En cada fila i, i=1...N, aparecen las instancias que realmente son de la clase i, mientras que las columnas j, j=1...N, son las que se han predicho al valor j de la clase. En el ejemplo anterior, la matriz de confusión que aparece es la siguiente:

```
=== Confusion Matrix ===

     a     b   <-- classified as
 18647     0 |     a = SI
     2   153 |     b = NO
```

por tanto, los valores en la diagonal son los aciertos, y el resto de valores son los errores. De los 18647 alumnos presentados, todos son correctamente clasificados, mientras que de los 155 no presentados, hay 153 correctamente clasificados y 2 con error.

Lista de resultados

Al igual que con otras opciones de análisis, la ventana izquierda de la lista de resultados contiene el resumen de todas las aplicaciones de clasificadores sobre conjuntos de datos en la sesión del *Explorer*. Puede accederse a esta

lista para presentar los resultados, y al activar el botón derecho aparecen diferentes opciones de visualización, entre las que podemos destacar las siguientes:

- Salvar y cargar modelos: **Load model, Save model**. Estos modelos pueden recuperarse de fichero para posteriormente aplicarlos a nuevos conjuntos de datos

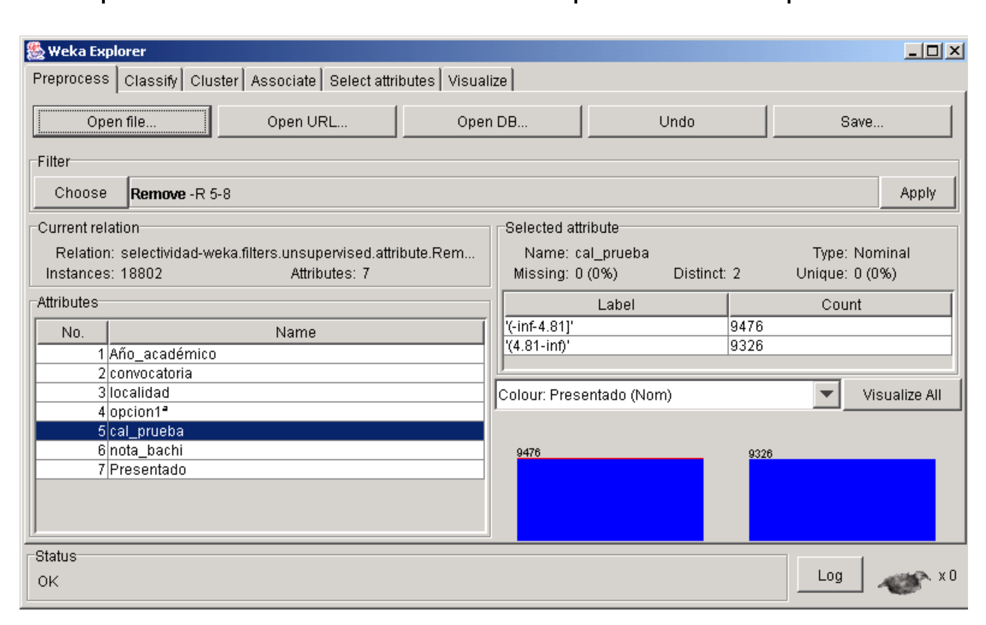
- Visualizar árbol y errores de predicción: **Visualize tree, Visualize classifier errors**,...

el árbol (permite almacenar Una vez se ejecuta el clasificador seleccionado sobre los datos de la relación,

Selección y configuración de clasificadores

Vamos a ilustrar la aplicación de algoritmos de clasificación a diferentes problemas de predicción de atributos definidos sobre los datos de entrada en este ejemplo. El problema de clasificación siempre se realiza sobre un atributo simbólico, en el caso de utilizar un atributo numérico se precisa por tanto discretizarlo antes en intervalos que representarán los valores de clase.

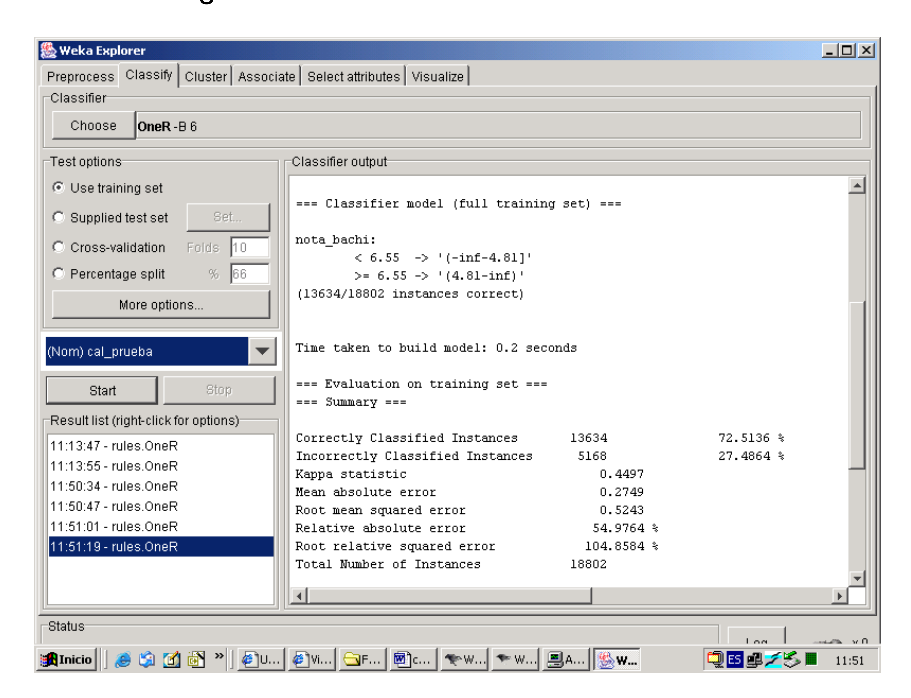
En primer lugar efectuaremos análisis de predicción de la calificación en la prueba de selectividad a partir de los siguientes atributos: año, convocatoria, localidad, opción, presentado y nota de bachillerato. Se van a realizar dos tipos de predicciones: aprobados, e intervalos de clasificación. Por tanto tenemos que aplicar en primer lugar una combinación de filtros que elimine los atributos no deseados relativos a calificaciones parciales y asignaturas opcionales, y un filtro que discretice la calificación en la prueba en dos partes:



obsérvese que se prefiere realizar las predicciones sobre la calificación en la prueba, puesto que la calificación final depende explícitamente de la nota del bachillerato.

Clasificador "OneR"

Este es uno de los clasificadores más sencillos y rápidos, aunque en ocasiones sus resultados son sorprendentemente buenos en comparación con algoritmos mucho más complejos. Simplemente selecciona el atributo que mejor "explica" la clase de salida. Si hay atributos numéricos, busca los umbrales para hacer reglas con mejor tasa de aciertos. Lo aplicaremos al problema de predicción de aprobados en la prueba a partir de los atributos de entrada, para llegar al resultado siguiente:



por tanto, el algoritmo llega a la conclusión que la mejor predicción posible con un solo atributo es la nota del bachillerato, fijando el umbral que determina el éxito en la prueba en 6.55. La tasa de aciertos sobre el propio conjunto de entrenamiento es del 72.5%. Compárese este resultado con el obtenido mediante ejecución sobre instancias independientes.

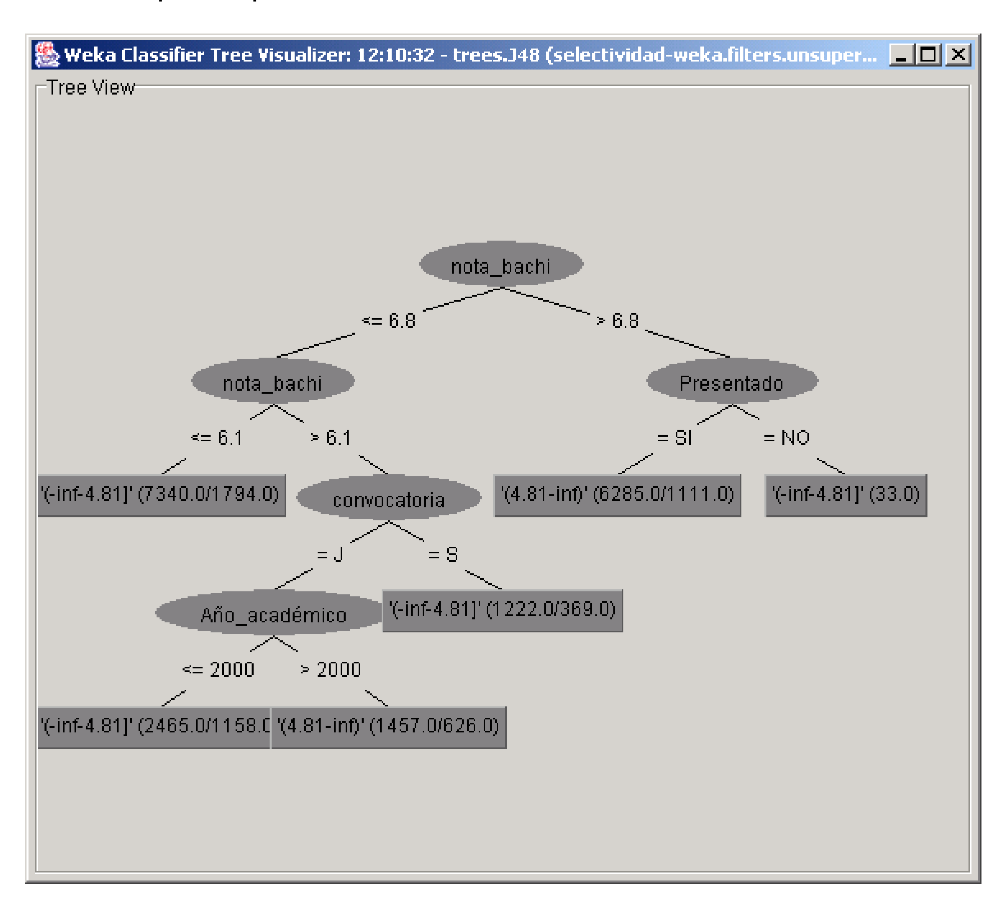
Clasificador como árbol de decisión: J48

El algoritmo J48 de WEKA es una implementación del algoritmo C4.5, uno de los algoritmos de minería de datos que más se ha utilizado en multitud de aplicaciones. No vamos a entrar en los detalles de todos los parámetros de configuración, dejándolo para el lector interesado en los detalles de este algoritmo, y únicamente resaltaremos uno de los más importantes, el factor de confianza para la poda, **confidence level**, puesto que influye notoriamente en el tamaño y capacidad de predicción del árbol construido.

Una explicación simplificada de este parámetro de construcción del árbol es la siguiente: para cada operación de poda, define la probabilidad de error que se permite a la hipótesis de que el empeoramiento debido a esta operación es significativo. Cuanto más baja se haga esa probabilidad, se exigirá que la diferencia en los errores de predicción antes y después de podar sea más

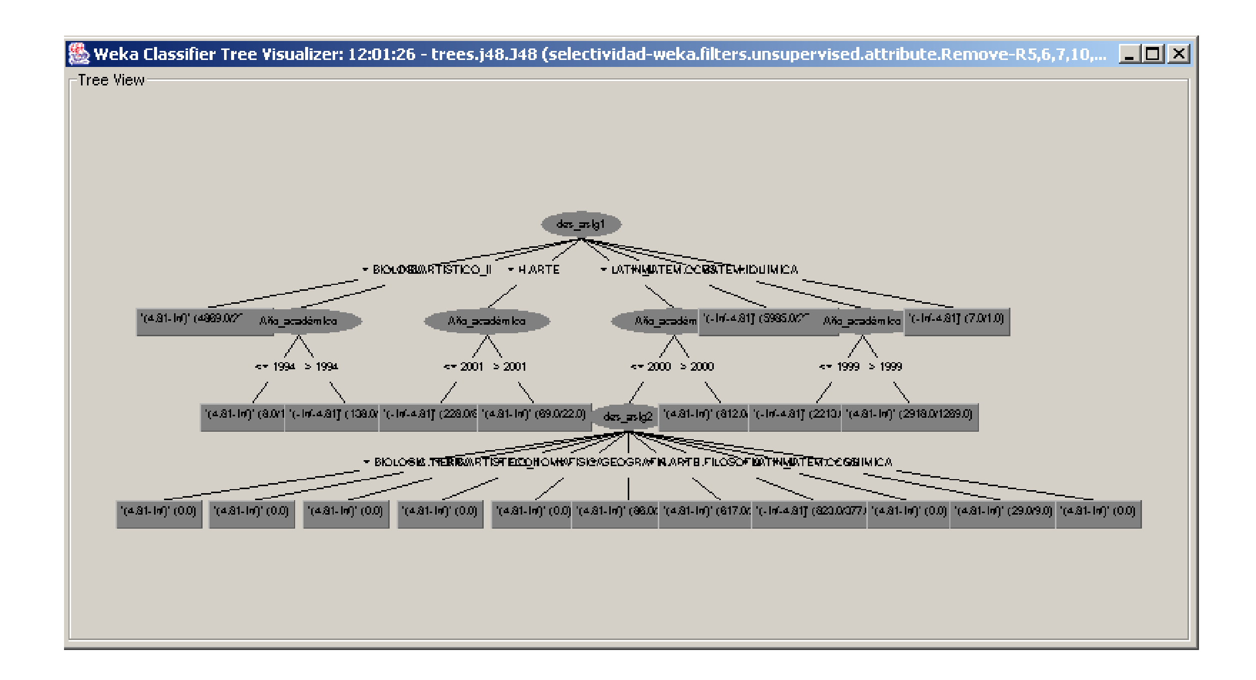
significativa para no podar. El valor por defecto de este factor es del 25%, y conforme va bajando se permiten más operaciones de poda y por tanto llegar a árboles cada vez más pequeños. Otra forma de variar el tamaño del árbol es a través de un parámetro que especifica el mínimo número de instancias por nodo, si bien es menos elegante puesto que depende del número absoluto de instancias en el conjunto de partida.

Construiremos el árbol de decisión con los parámetros por defecto del algoritmo J48: se llega a un clasificador con más de 250 nodos, con una probabilidad de acierto ligeramente superior al del clasificador *OneR*. Modifique ahora la configuración del algoritmo para llegar a un árbol más manejable, como el que se presenta a continuación



Obsérvese que este modelo es un refinamiento del generado con OneR, que supone una mejorar moderada en las prestaciones. De nuevo los atributos más importantes son la calificación de bachillerato, la convocatoria, y después el año, antes que la localidad o las opciones. Analice las diferencias con evaluación independiente y validación cruzada, y compárelas con las del árbol más complejo con menos poda.

Podría ser de interés analizar el efecto de las opciones y asignaturas seleccionadas sobre el éxito en la prueba, para lo cual quitaremos el atributo más importante, nota de bachillerato. Llegamos a un árbol como el siguiente, en el que lo más importante es la primera asignatura optativa, en diferentes combinaciones con el año y segunda asignatura optativa:

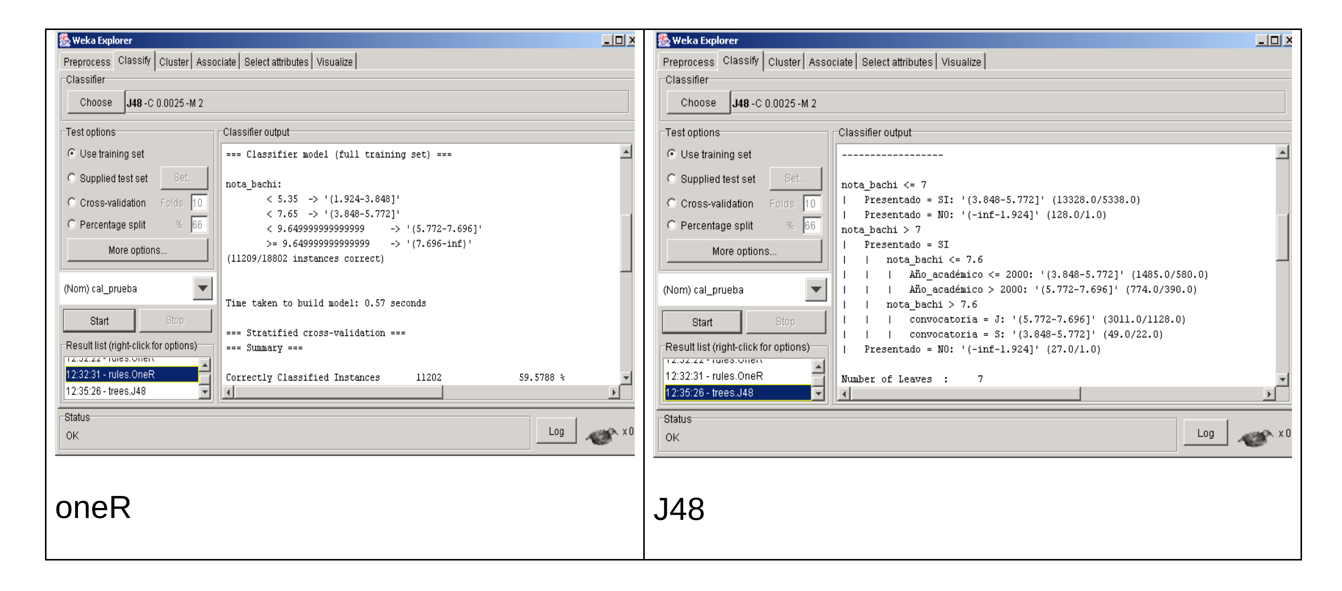


Este resultado generado por el clasificador puede comprobarse si se analizan los histogramas de cada variable y visualizando el porcentaje de aprobados con el color, que esta variable es la que mejor separa las clases, no obstante, la precisión apenas supera el 55%.

Otros problemas de clasificación pueden formularse sobre cualquier atributo de interés, a continuación mostramos algunos ejemplos a título ilustrativo.

Clasifiación multinivel de las calificaciones

el problema anterior puede intentar refinarse y dividir el atributo de interés, la calificación final, en más niveles, en este caso 5. Los resultados se muestran a continuación

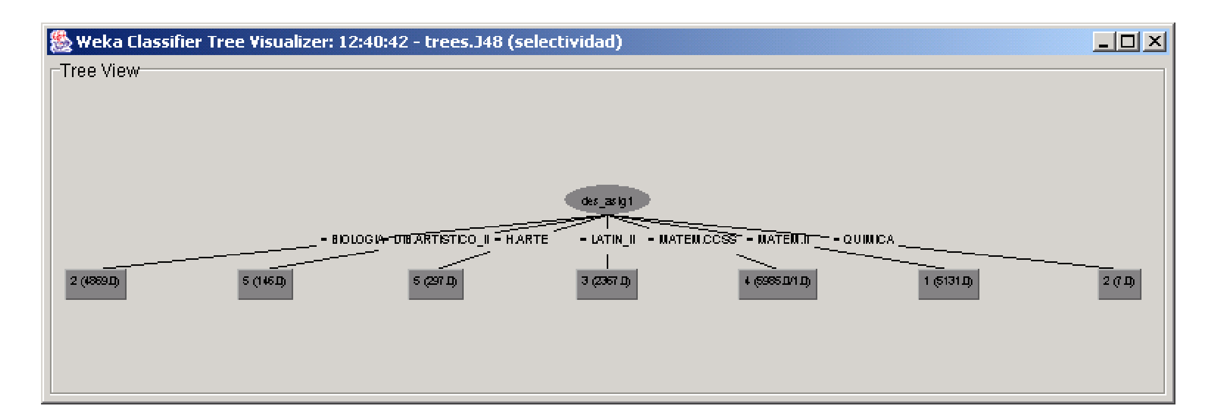


oneR	J48

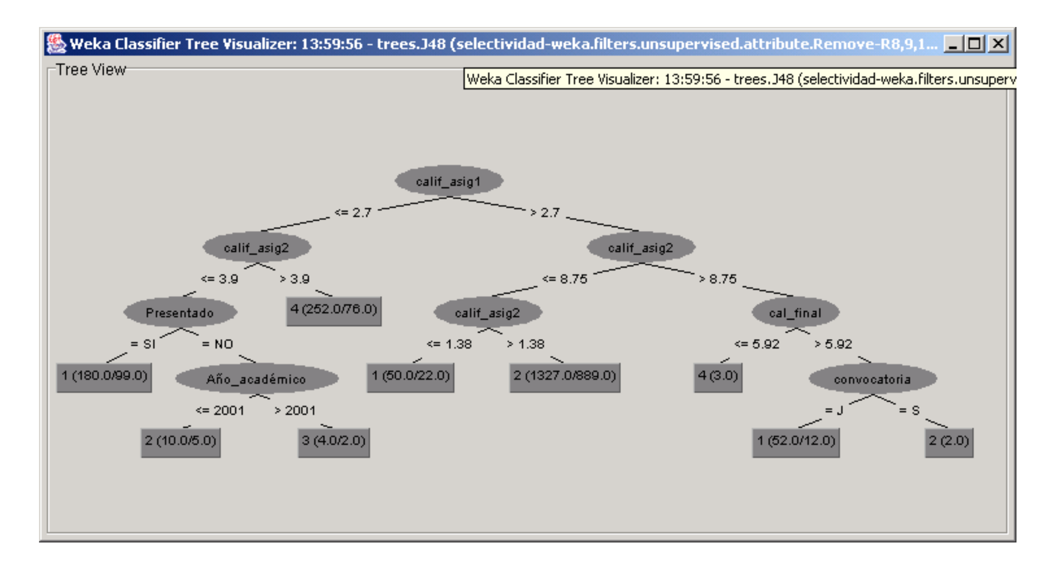
La precisión alcanzada es tan sólo del 60%, indicando que hay bastante incertidumbre una vez generada la predicción con los modelos anteriores.

Predicción de la opción

Si dejamos todos los atributos en la muestra y aplicamos el clasificador a la opción cursado, se desvela una relación trivial entre opción y asignaturas en las opciones que predice con prácticamente el 100% de los casos.

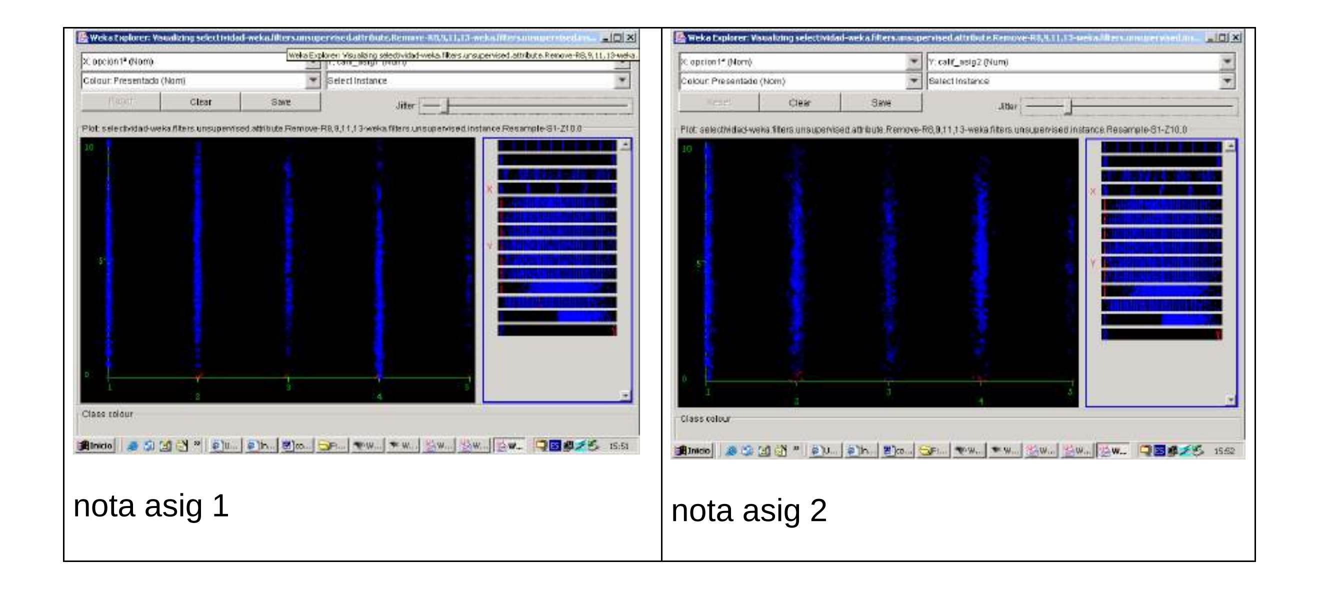


A continuación eliminamos estos designadores con un filtro de atributos. Si aplicamos el algoritmo J48 sobre los datos filtrados, llegamos a un árbol de más de 400 nodos, y con muchísimo sobre-ajuste (observe la diferencia de error de predicción sobre el conjunto de entrenamiento y sobre un conjunto independiente). Forzando la poda del árbol, llegamos al modelo siguiente:

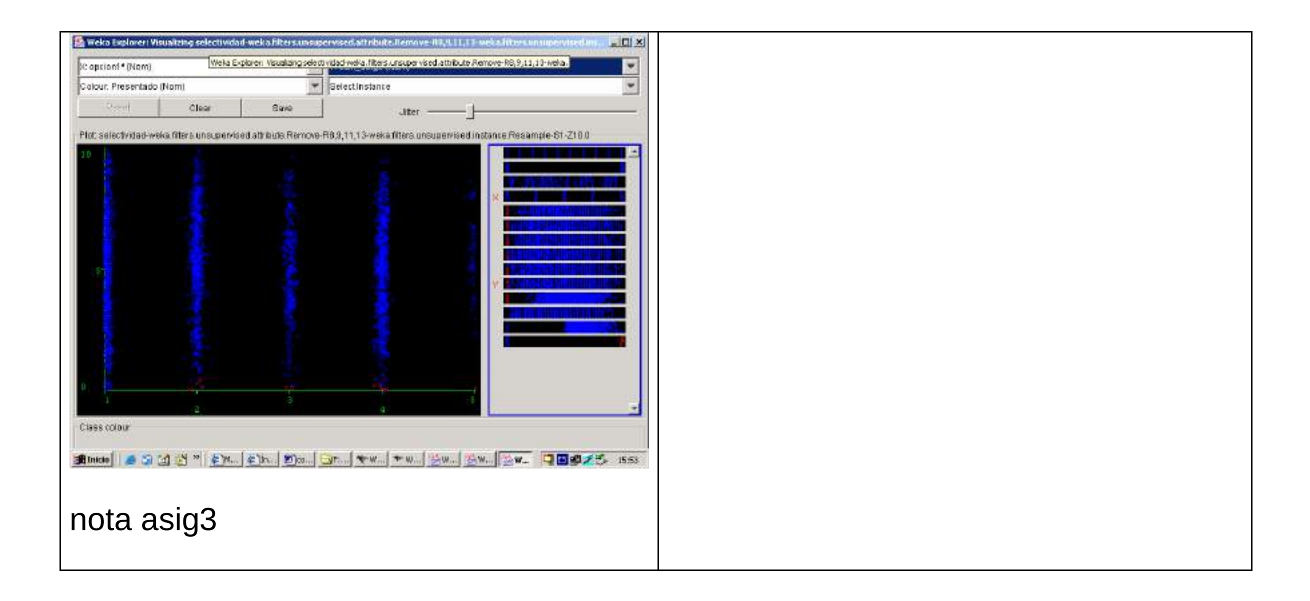


los atributos más significativos para separar las opciones son precisamente las calificaciones en las asignaturas optativas, pero apenas predice correctamente un 40% de los casos. Por tanto, vemos que no hay una relación directa entre opciones y calificaciones en la prueba, al menos relaciones que se puedan modelar con los algoritmos de clasificación disponibles. Si nos fijamos en detalle en las calificaciones en función de las opciones, podríamos determinar que apenas aparecen diferencias aparecen en los últimos percentiles, a la vista de las gráficas siguientes:

nota historia	nota idioma
nota lengua	nota final



nota asig 1 nota asig 2

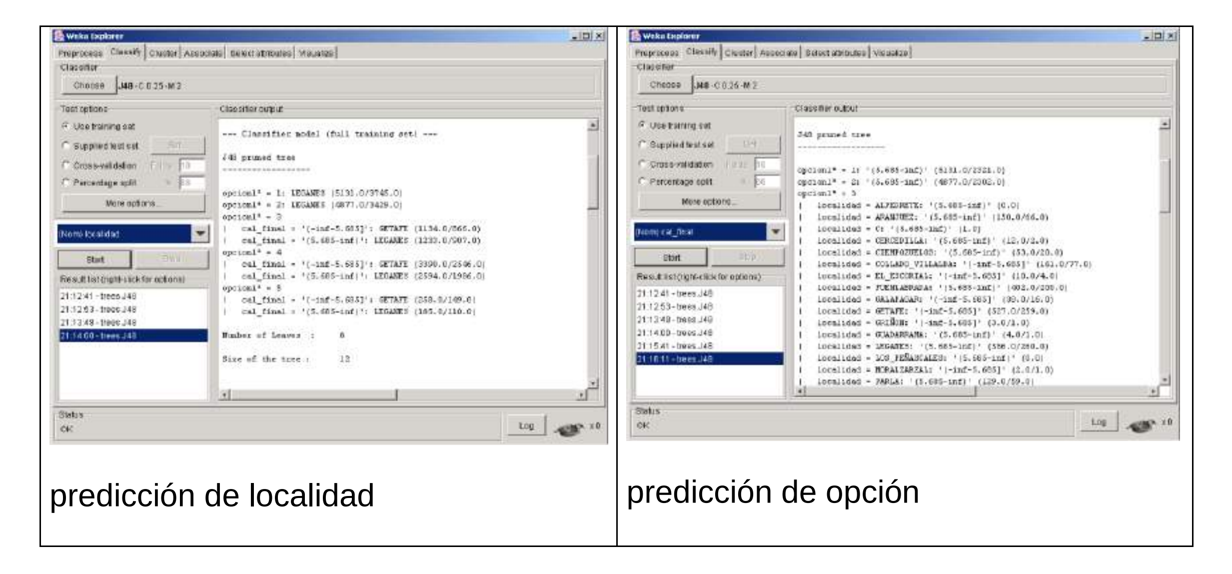


nota asig3

Vemos que las diferencias no son significativas, salvo quizá en los últimos percentiles.

Predicción de localidad y opción

La clasificación se puede realizar sobre cualquier atributo disponible. Con el número de atributos reducido a tres, localidad, opción y calificación (aprobados y suspensos), vamos a buscar modelos de clasificación, para cada uno de los atributos:



predicción de localidad

predicción de opción

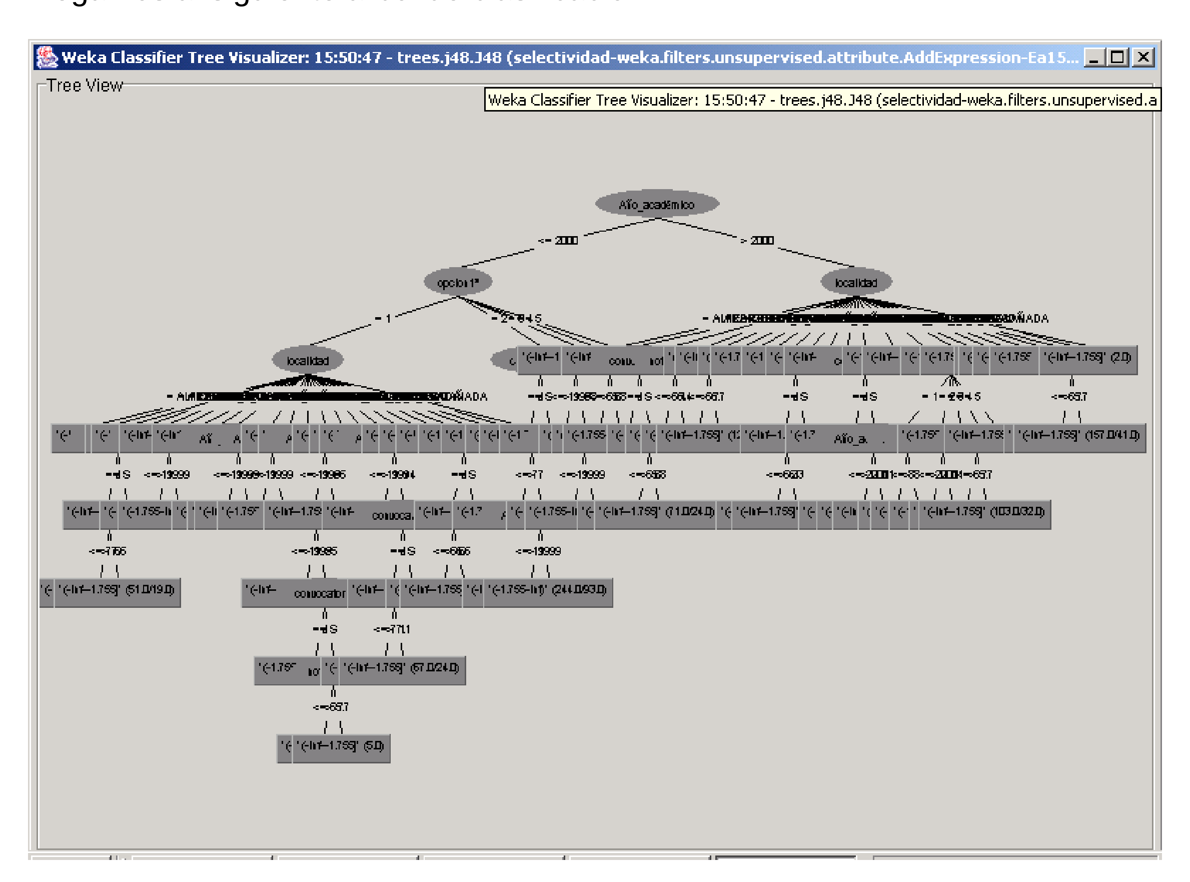
Es decir, la opción 1 y 2 aparecen mayoritariamente en Leganés, y las opciones 3 y 4 más en los alumnos que aprobaron la prueba en Leganés. No obstante, obsérvese que los errores son tan abrumadores (menos del 30% de aciertos) que cuestionan fuertemente la validez de estos modelos.

Mejora en la prueba

Un problema de clasificación interesante puede ser determinar qué alumnos tienen más "éxito" en la prueba, en el sentido de mejorar su calificación de bachillerato con la calificación en la prueba. Para ello utilizaremos el atributo "mejora", introducido en la sección 1.4.2.3, y lo discretizamos en dos valores de la misma frecuencia (obtenemos una mediana de -1.75, de manera que dividimos los alumnos en dos grupos: los que obtienen una diferencia menor a este valor y superior a este valor, para diferenciar los alumnos según el resultado se atenga más o menos a sus expectativas. Evidentemente, para evitar construir modelos triviales, tenemos que eliminar los atributos relacionados con las calificaciones en la prueba, para no llegar a la relación que acabamos de construir entre la variable calculada y las originales. Vamos a preparar el problema de clasificación con los siguientes atributos:

Attributes: 7
Año_académico
convocatoria
localidad
opcion1ª
nota_bachi
Presentado
mejora

Llegamos al siguiente árbol de clasificación.



Es decir, los atributos que más determinan el "éxito" en la prueba son: año académico, opción y localidad. Para estos resultados tenemos una precisión, con evaluación sobre un conjunto independiente, en torno al 60%, por lo que sí podríamos tomarlo en consideración.

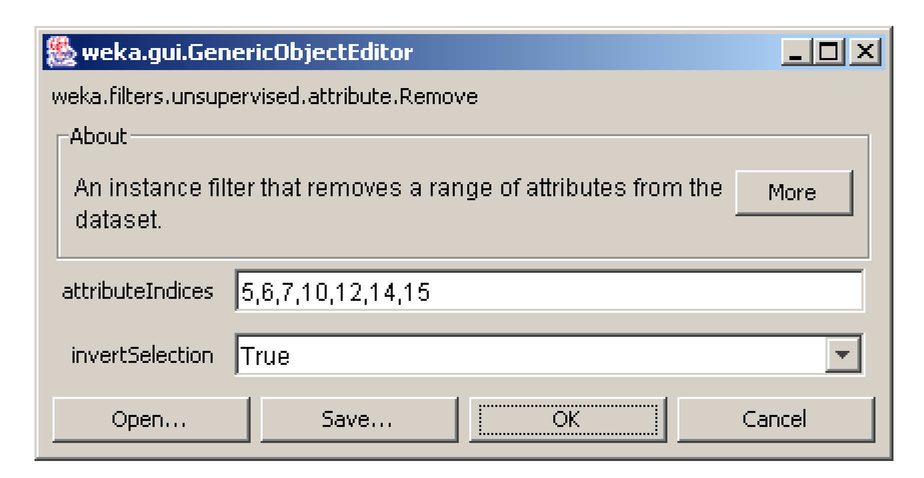
Predicción numérica

La predicción numérica se define en WEKA como un caso particular de clasificación, en el que la clase es un valor numérico. No obstante, los algoritmos integrados para clasificar sólo admiten clases simbólicas y los algoritmos de predicción numéricas, que aparecen mayoritariamente en el apartado *classifiers->functions*, aunque también en *classifiers->trees*.

Vamos a ilustrar algoritmos de predicción numérica en WEKA con dos tipos de problemas. Por un lado, "descubrir" relaciones deterministas que aparecen entre variables conocidas, como calificación en la prueba con respecto a las parciales y la calificación final con respecto a la prueba y bachillerato, y buscar otros modelos de mayor posible interés.

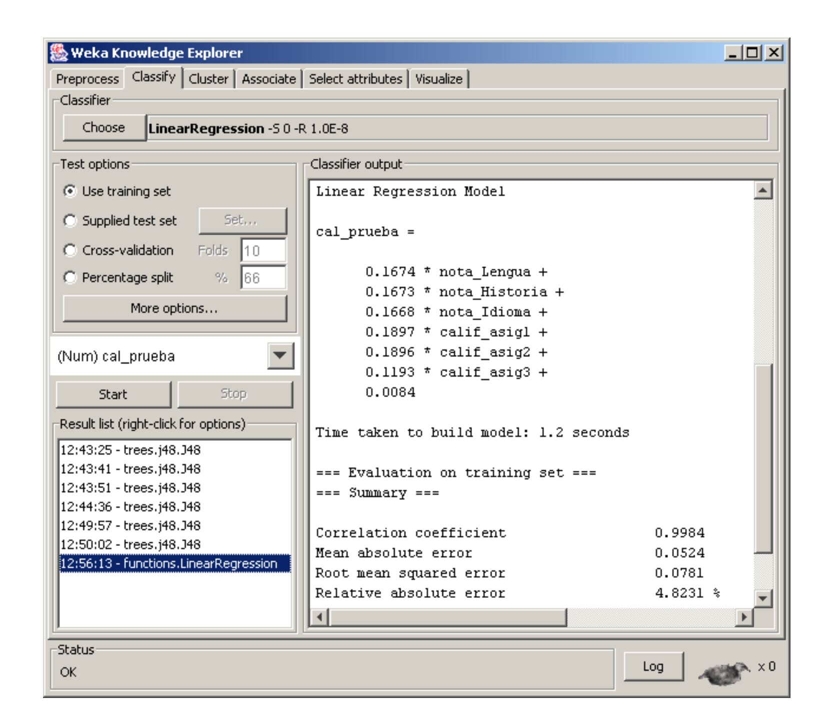
Relación entre calificación final y parciales

Seleccionamos los atributos con las 6 calificaciones parciales y la calificación en la prueba:



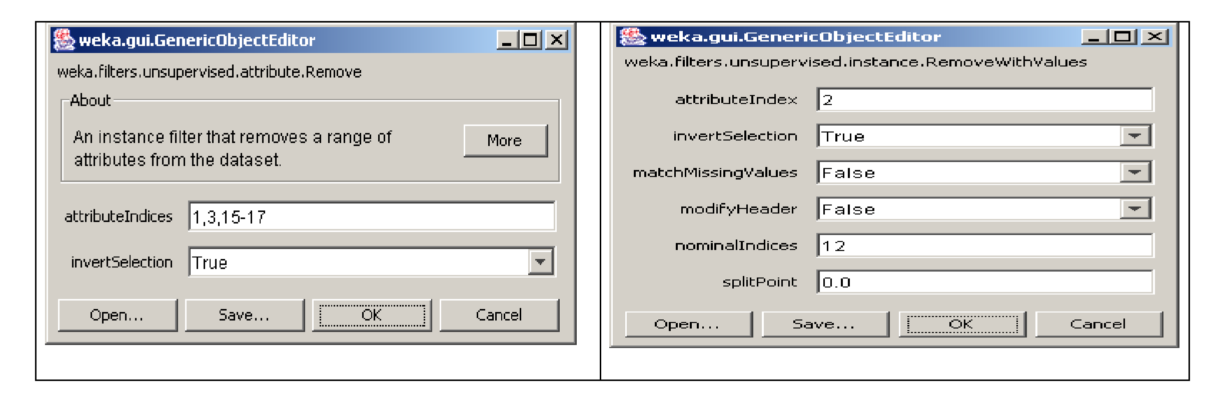
Vamos a aplicar el modelo de predicción más popular: regresión simple, que construye un modelo lineal del atributo clase a partir de los atributos de entrada: **functions->LinearRegresion**

Como resultado, aparece la relación con los pesos relativos de las pruebas parciales sobre la calificación de la prueba:

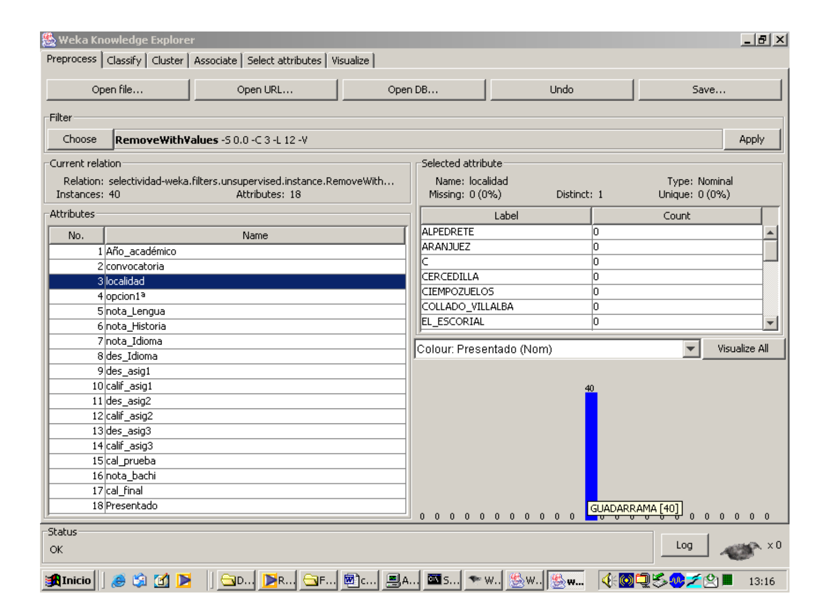


Hay que observar que en los problemas de predicción la evaluación cambia, apareciendo ahora el coeficiente de correlación y los errores medio y medio cuadrático, en términos absolutos y relativos. En este caso el coeficiente de correlación es de 0.998, lo que indica que la relación es de una precisión muy notable.

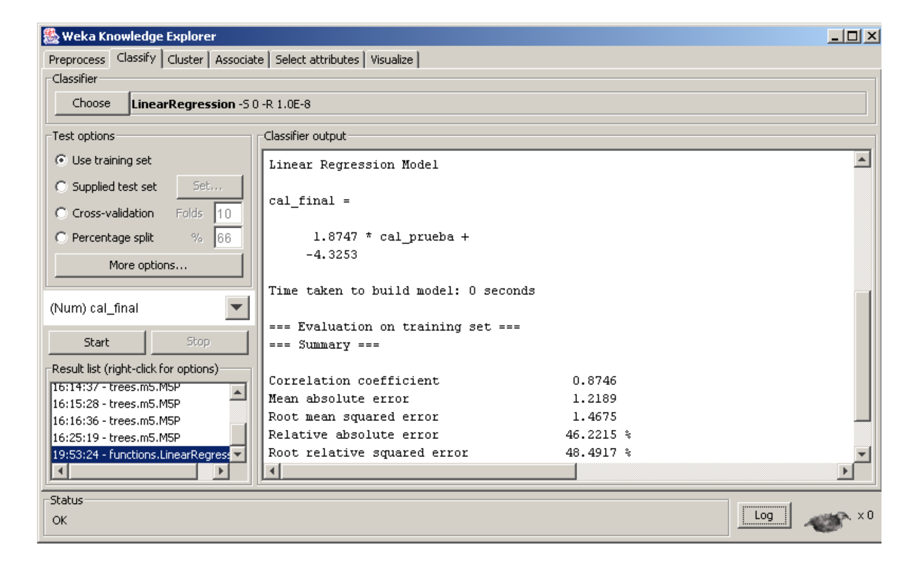
Si aplicamos ahora esta función a la relación entre calificación final con calificación en la prueba y nota de bachillerato (filtro que selecciona únicamente los atributos 15-17), podemos determinar la relación entre estas variables: qué peso se lleva la calificación de bachillerato y de la prueba en la nota final. Vamos a hacerlo primero con los alumnos de una población pequeña, de Guadarrama (posición 12 del atributo localidad). Aplicamos los filtros correspondientes para tener únicamente estos alumnos, y los atributos de calificaciones de la prueba, bachillerato y final:



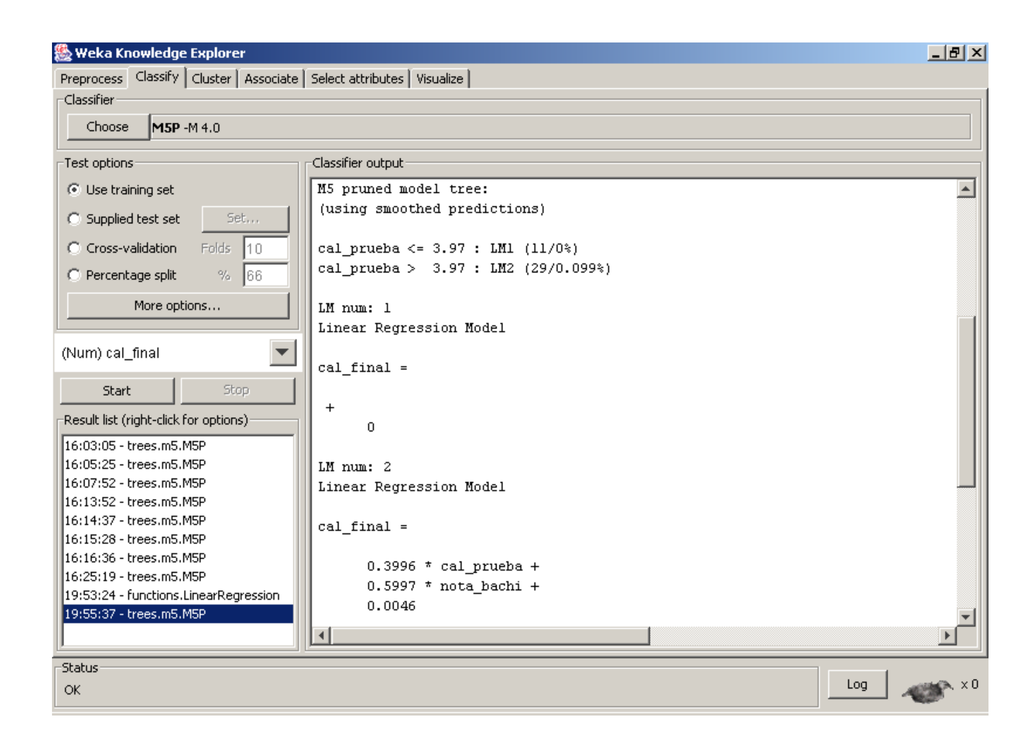
llegamos a 40 instancias:



si aplicáramos regresión lineal como en el ejemplo anterior, obtenemos el siguiente resultado:

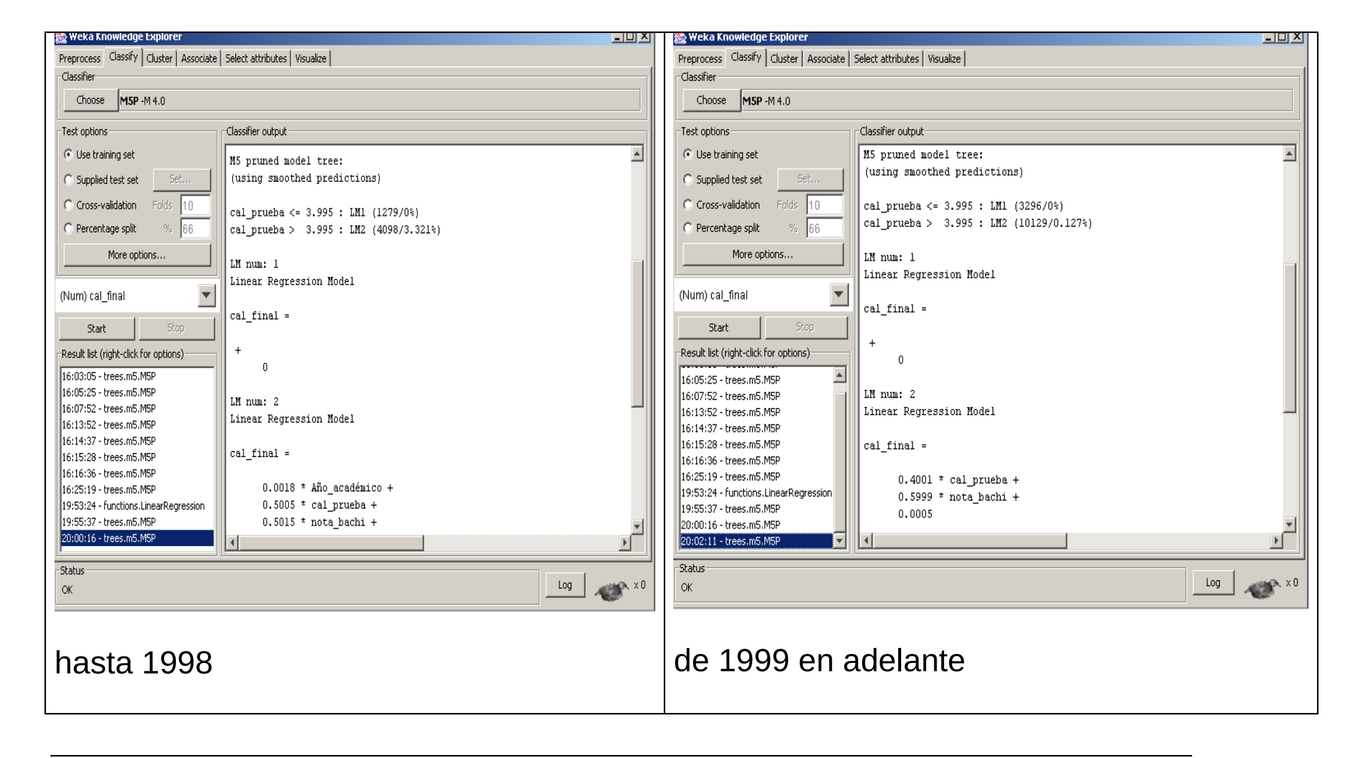


el resultado deja bastante que desear porque la relación no es lineal. Para solventarlo podemos aplicar el algoritmo M5P, seleccionado en WEKA como **trees->m5->M5P,** que lleva a cabo una regresión por tramos, con cada tramo determinado a partir de un árbol de regresión. Llegamos al siguiente resultado:



que es prácticamente la relación exacta utilizada en la actualidad: 60% nota de bachillerato y 40% de la prueba, siempre que se supere en ésta un valor mínimo de 4 puntos.

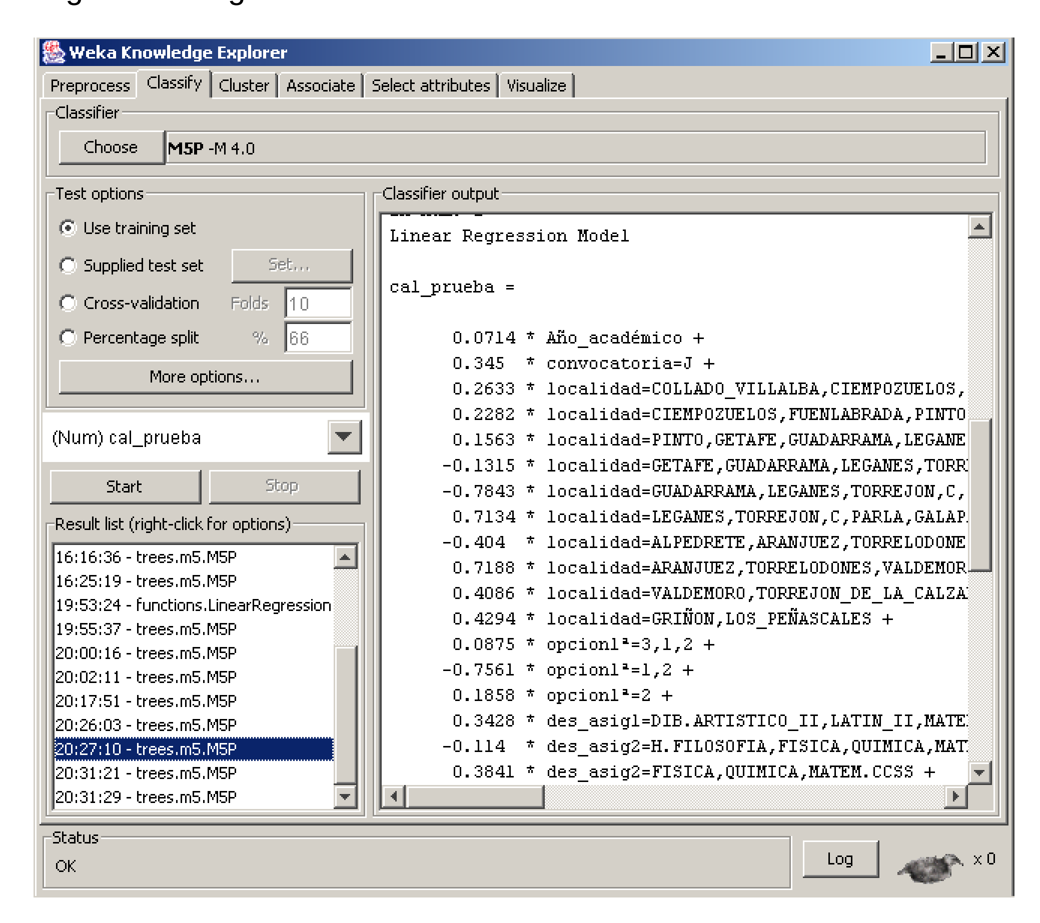
Si aplicamos este algoritmo a otros centros no siempre obtenemos este resultado, por una razón: hasta 1998 se ponderaba al 50%, y a partir de 1999 se comenzó con la ponderación anterior. Verifíquese aplicando este algoritmo sobre datos filtrados que contengan alumnos de antes de 1998 y de 1999 en adelante. En este caso, el algoritmo M5P no tiene capacidad para construir el modelo correcto, debido a la ligera diferencia en los resultados al cambiar la forma de ponderación. Los árboles obtenidos en ambos casos se incluyen a continuación:



hasta 1998	de 1999 en adelante

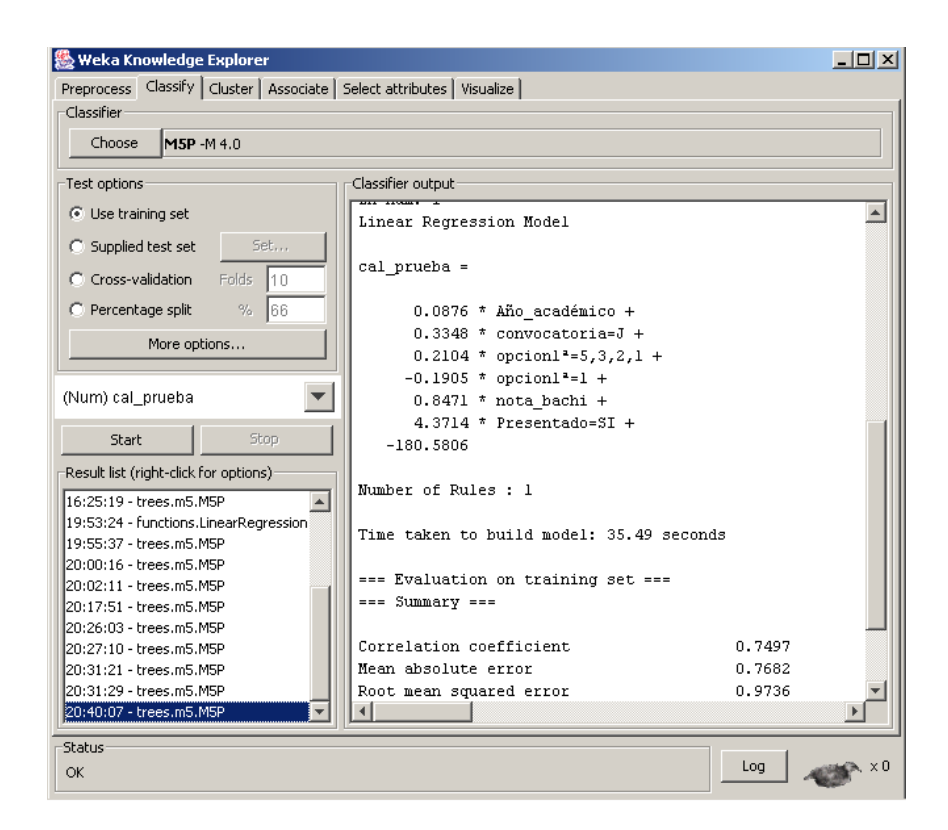
Predicción de la calificación

Vamos a aplicar ahora este modelo para intentar construir un modelo aplicación más interesante, o, al menos, analizar tendencias de interés. Se trata de intentar predecir la calificación final a partir de los atributos de entrada, los mismos que utilizamos para el problema de clasificar los alumnos que aprueban la prueba. Si aplicamos el algoritmo sobre el conjunto completo llegamos al siguiente modelo:

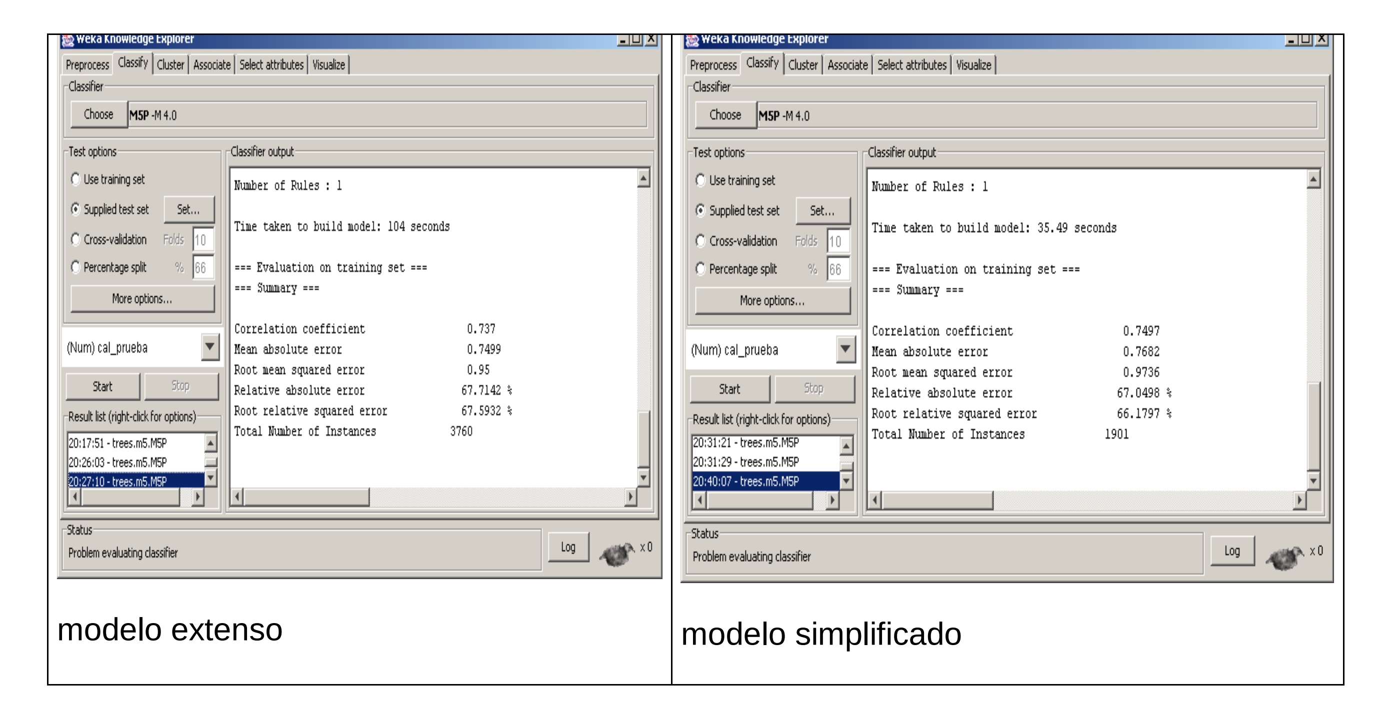


obsérvese cómo trata el algoritmo los atributos nominales para incluirlos en la regresión: ordena los valores según el valor de la magnitud a predecir (en el caso de localidad, desde Collado hasta Los Peñascales y en el de opción, ordenadas como 4º, 5º, 3º, 2º, 1º), y va tomando variables binarias resultado de dividir en diferentes puntos, determinando su peso en la función. En esta función lo que más pesa es la convocatoria, después la nota de bachillerato, y después entran en juego la localidad, asignaturas optativas, y opción, con un modelo muy complejo.

Si simplificamos el conjunto de atributos de entrada, y nos quedamos únicamente con el año, opción, nota de bachillerato, y convocatoria, llegamos a:



este modelo es mucho más manejable. Compare los errores de predicción con ambos casos:

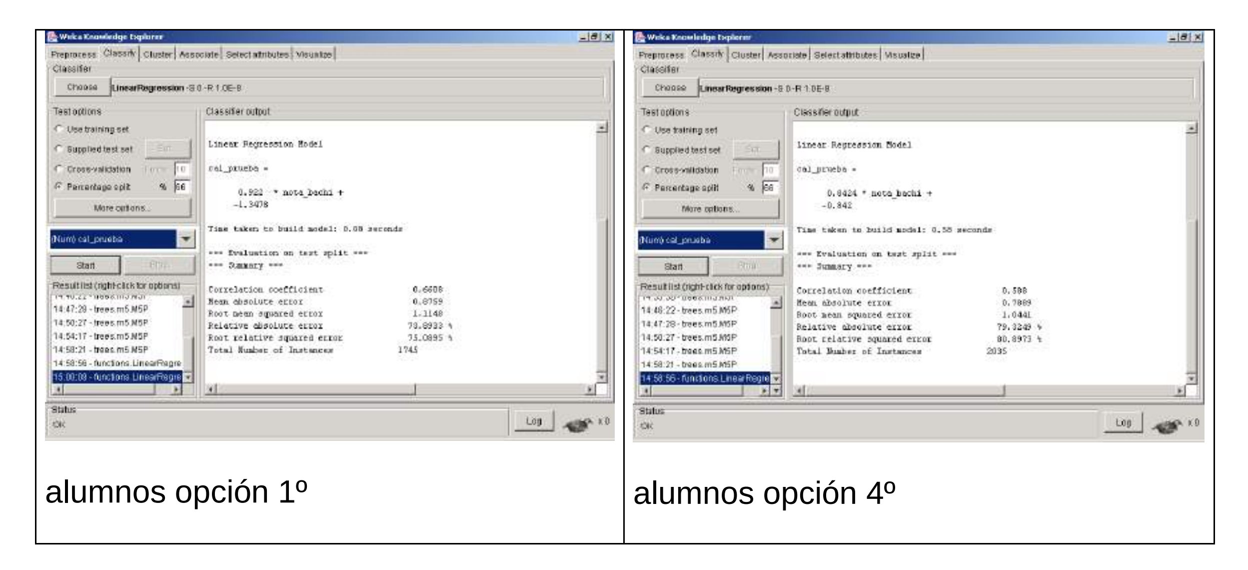


modelo extenso | modelo simplificado

Correlación entre nota de bachillerato y calificación en prueba

Finalmente, es interesante a veces hacer un modelo únicamente entre dos variables para ver el grado de correlación entre ambas. Continuando con nuestro interés por las relaciones entre calificación en prueba y calificación en bachillerato, vamos a ver las diferencias por opción. Para ello filtraremos por un lado los alumnos de opción 1 y los de opción 4. A continuación dejamos

únicamente los atributos calificación en prueba y nota de bachillerato, para analizar la correlación de los modelos para cada caso.



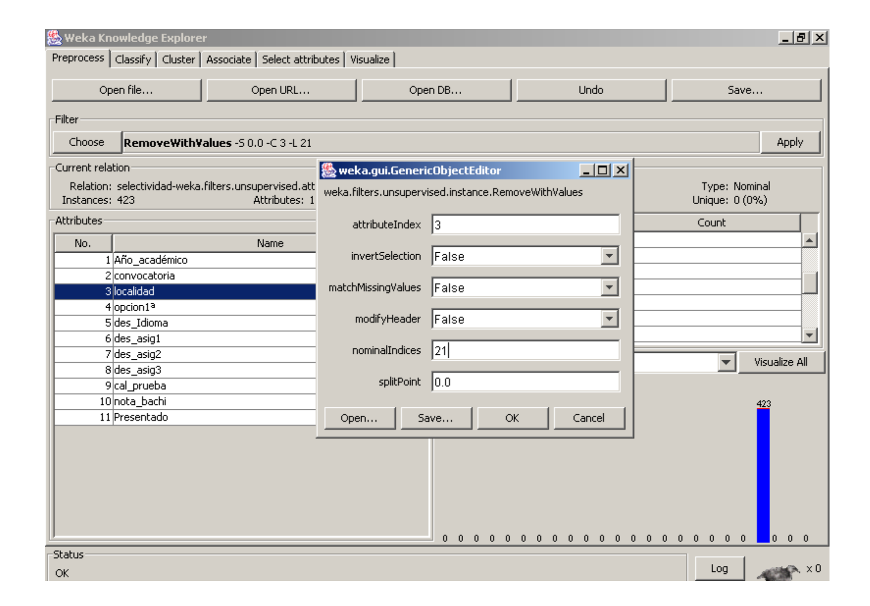
alumnos opción 1º	alumnos opción 4º

podemos concluir que para estas dos opciones el grado de relación entre las variables sí es significativamente diferente, los alumnos que cursan la opción 1º tienen una relación más "lineal" entre ambas calificaciones que los procedentes de la opción 4º

Aprendizaje del modelo y aplicación a nuevos datos.

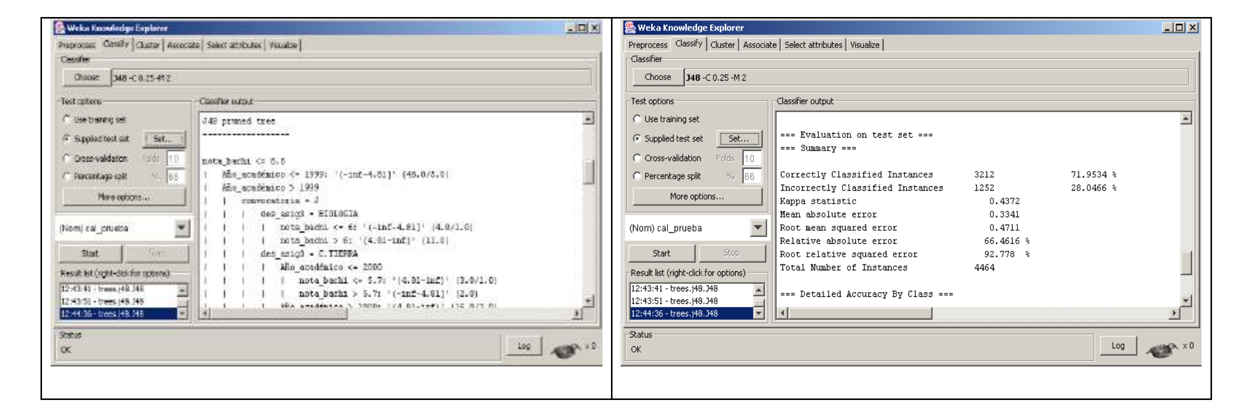
Para finalizar esta sección de clasificación, ilustramos aquí las posibilidades de construir y evaluar un clasificador de forma cruzada con dos ficheros de datos. Seleccionaremos el conjunto atributos siguiente: Año_académico, convocatoria, localidad, opcion1ª, des_Idioma, des_asig1, des_asig2, des_asig3, cal_prueba, nota_bachi, Presentado. El atributo con la calificación, “cal_prueba”, lo discretizamos en dos intervalos.

Vamos a generar, con el filtro de instancias dos conjuntos de datos correspondientes a los alumnos de Getafe y Torrelodones. Para ello primero seleccionamos las instancias con el atributo localidad con valor 10, lo salvamos (“datosGetafe”) y a continuación las instancias con dicho atributo con valor 21 (“datosTorrelodones”).

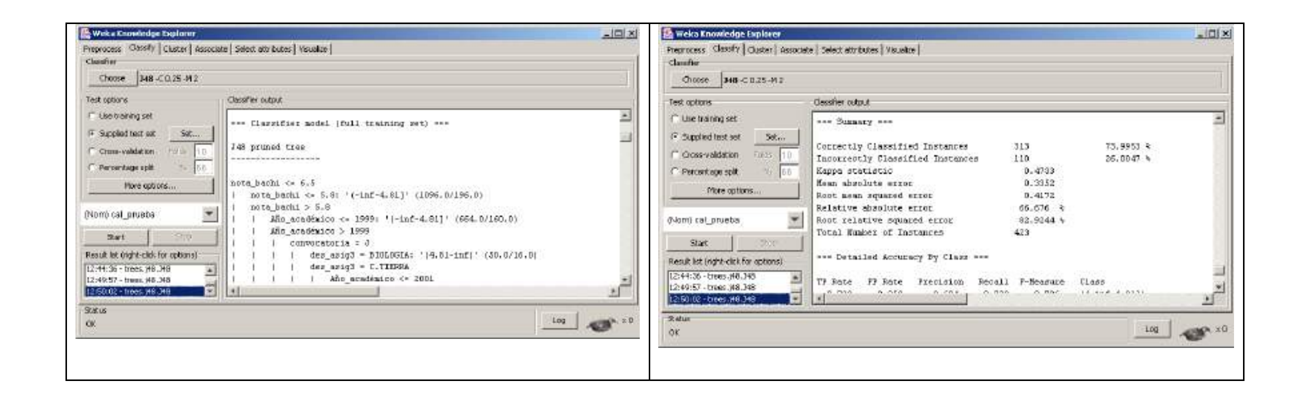


Ahora vamos a generar los modelos de clasificación de alumnos con buen y mal resultado en la prueba con el fichero de alumnos de la localidad de Torrelodones, para evaluarlo con los alumnos de Getafe.

Para ello en primer lugar cargamos el fichero con los alumnos de Torrelodones que acabamos de generar, “datosTorrelodones”, y lo evaluamos sobre el conjunto con alumnos de Getafe. Para ello, seleccionaremos la opción de evaluación con un fichero de datos independiente, **Supplied test set**, y fijamos con el botón **Set,** que el fichero de test es “datosGetafe”. Obsérvese el modelo generado y los resultados:



Si ahora hacemos la operación inversa, entrenar con los datos de Getafe y evaluar con los de Torrelodones, llegamos a:



Hay ligeras diferencias en los modelos generados para ambos conjuntos de datos (para los alumnos de Torrelodones, lo más importante es tener una calificación de bachillerato superior a 6.8, mientras que a los de Getafe les basta con un 6.5), y los resultados de evaluación con los datos cruzados muestran una variación muy pequeña. El modelo construido a partir de los datos de Torrelodones predice ligeramente peor los resultados de Getafe que a la inversa.

Selección de atributos

Esta última sección permite automatizar la búsqueda de subconjuntos de atributos más apropiados para "explicar" un atributo objetivo, en un sentido de clasificación supervisada: permite explorar qué subconjuntos de atributos son los que mejor pueden clasificar la clase de la instancia. Esta selección "supervisada" aparece en contraposición a los filtros de preprocesado comentados en la sección 1.4.2, que se realizan de forma independiente al proceso posterior, razón por la que se etiquetaron como "no supervisados".

La selección supervisada de atributos tiene dos componentes:

- Método de Evaluación (**Attribute Evaluator**): es la función que determina la calidad del conjunto de atributos para discriminar la clase.
- Método de Búsqueda (**Search Method**): es la forma de realizar la búsqueda de conjuntos. Como la evaluación exhaustiva de todos los subconjuntos es un problema combinatorio inabordable en cuanto crece el número de atributos, aparecen estrategias que permiten realizar la búsqueda de forma eficiente

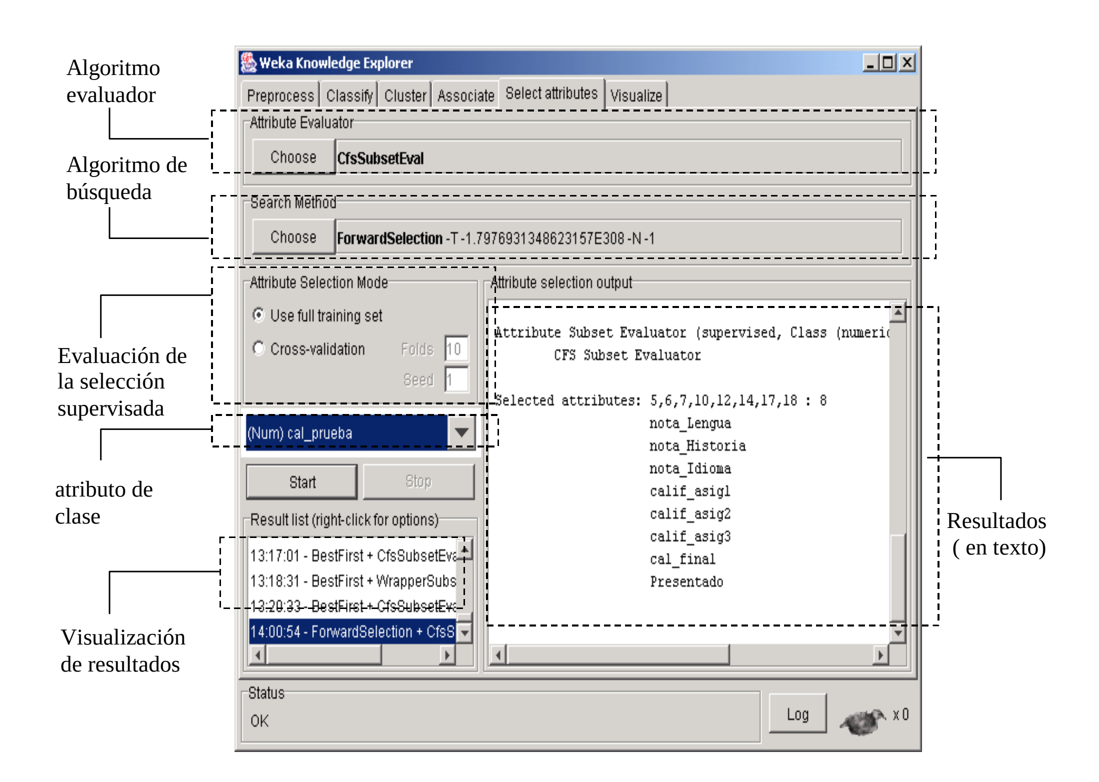
De los métodos de evaluación, podemos distinguir dos tipos: los métodos que directamente utilizan un clasificador específico para medir la calidad del subconjunto de atributos a través de la tasa de error del clasificador, y los que no. Los primeros, denominados métodos "wrapper", porque "envuelven" al clasificador para explorar la mejor selección de atributos que optimiza sus prestaciones, son muy costosos porque necesitan un proceso completo de entrenamiento y evaluación en cada paso de búsqueda. Entre los segundos podemos destacar el método **"CfsSubsetEval"**, que calcula la correlación de la

clase con cada atributo, y eliminan atributos que tienen una correlación muy alta como atributos redundantes.

En cuanto el método de búsqueda, vamos a mencionar por su rapidez el "**ForwardSelection**", que es un método de búsqueda subóptima en escalada, donde elije primero el mejor atributo, después añade el siguiente atributo que más aporta y continua así hasta llegar a la situación en la que añadir un nuevo atributo empeora la situación. Otro método a destacar sería el "**BestSearch"**, que permite buscar interacciones entre atributos más complejas que el análisis incremental anterior. Este método va analizando lo que mejora y empeora un grupo de atributos al añadir elementos, con la posibilidad de hacer retrocesos para explorar con más detalle. El método **"ExhaustiveSearch"** simplemente enumera todas las posibilidades y las evalúa para seleccionar la mejor

Por otro lado, en la configuración del problema debemos seleccionar qué atributo objetivo se utiliza para la selección supervisada, en la ventana de selección, y determinar si la evaluación se realizará con todas las instancias disponibles, o mediante validación cruzada.

Los elementos por tanto a configurar en esta sección se resumen en la figura siguiente:



Siguiendo con nuestro ejemplo, vamos a aplicar búsqueda de atributos para "explicar" algunos atributos objetivo. Para obtener resultados sin necesidad de mucho tiempo, vamos a seleccionar los algoritmos más eficientes de evaluación y búsqueda, **CsfSubsetEval** y **ForwardSelection**

Por ejemplo, para la calificación final tenemos 8 atributos seleccionados:

```
Selected attributes: 5,6,7,10,12,14,17,18 : 8
                     nota_Lengua
                     nota_Historia
                     nota_Idioma
                     calif_asig1
                     calif_asig2
                     calif_asig3
                     cal_final
                     Presentado
```

y para la opción 1 atributo:

```
Selected attributes: 9 : 1

                     des_asig1
```

Por tanto, hemos llegado a los atributos que mejor explican ambos (la calificación en la prueba depende directamente de las parciales, y la opción se explica con la 1ª asignatura), si bien son relaciones bastante triviales. A continuación preparamos los datos para buscar relaciones no conocidas, quitando los atributos referentes a cada prueba parcial. Dejando como atributos de la relación:

```
Attributes:    7
               Año_académico
               convocatoria
               localidad
               opcion1ª
               cal_prueba
               nota_bachi
               Presentado
```

para la calificación final llegamos a 2 atributos:

```
Selected attributes: 6,7 : 2
                     nota_bachi
                     Presentado
```

y para la opción 2:

```
Selected attributes: 3,5,6 : 3
                     localidad
                     cal_prueba
                     nota_bachi
```

No obstante, si observamos la figura de mérito con ambos problemas, que aparece en la ventana textual de resultados, vemos que este segundo es mucho menos fiable, como ya hemos comprobado en secciones anteriores.

Capítulo 5. Implementación de las técnicas de Análisis de Datos en Weka

5.1. Utilización de las clases de WEKA en programas independientes

5.2. Tabla de Decisión en WEKA

El algoritmo de tabla de decisión implementado en la herramienta WEKA se encuentra en la clase *weka.classifiers.DecisionTable.java*. Las opciones de configuración de que disponen son las que vemos en la tabla 5.1.

Tabla 5.1: *Opciones de configuración para el algoritmo de tabla de decisión en WEKA.*

Opción	***Descripción***
useIBk (False)	Utilizar NN (ver punto 2.2.5.1) en lugar de la tabla de decisión si no la instancia a clasificar no se corresponde con ninguna regla de la tabla.
displayRules (False)	Por defecto no se muestran las reglas del clasificador, concretamente la tabla de decisión construida.
maxStale (5)	Indica el número máximo de conjuntos que intenta mejorar el algoritmo para encontrar una tabla mejor sin haberla encontrado en los últimos *n-1* subconjuntos.
crossVal (1)	Por defecto se evalúa el sistema mediante el proceso *leave-one-out*. Si se aumenta el valor *1* se realiza validación cruzada con *n* carpetas.

En primer lugar, en cuanto a los atributos que permite el sistema, éstos pueden ser tanto numéricos (que se discretizarán) como simbólicos. La clase también puede ser numérica o simbólica.

El algoritmo consiste en ir seleccionando uno a uno los subconjuntos, añadiendo a cada uno de los ya probados cada uno de los atributos que aún no pertenecen a él. Se prueba la precisión del subconjunto, bien mediante validación cruzada o *leave-one-out* y, si es mejor, se continúa con él. Se continúa así hasta que se alcanza *maxStale*. Para ello, una variable comienza siendo *0* y aumenta su valor en una unidad cuando a un subconjunto no se le puede añadir ningún atributo para mejorarlo, volviendo a *0* si se añade un nuevo atributo a un subconjunto.

En cuanto al proceso *leave-one-out*, es un método de estimación del error. Es una validación cruzada en la que el número de conjuntos es igual al número de ejemplos de entrenamiento. Cada vez se elimina un ejemplo del conjunto de entrenamiento y se entrena con el resto. Se juzgará el acierto del sistema con el resto de instancias según se acierte o se falle en la predicción del ejemplo que se eliminó. El resultado de las *n* pruebas (siendo *n* el número inicial de ejemplos de entrenamiento) se promedia y dicha media será el error estimado.

Por último, para clasificar un ejemplo pueden ocurrir dos cosas. En primer lugar, que el ejemplo corresponda exactamente con una de las reglas de la tabla de decisión, en cuyo caso se devolverá la clase de dicha regla. Si no se corresponde con ninguna regla, se puede utilizar *Ibk* (si se seleccionó dicha opción) para predecir la clase, o la media o moda de la clase según el tipo de clase del que se trate (numérica o simbólica).

5.3. ID3 en WEKA

La clase en la que está codificado el algoritmo ID3 es *weka.classifiers.ID3.java*. En primer lugar, en cuanto a la implementación, no permite ningún tipo de configuración. Esta implementación se ajusta exactamente a lo descrito anteriormente. Lo único reseñable es que para determinar si un nodo es hoja o no, se calcula la ganancia de información y, si la máxima ganancia es *0* se considera nodo hoja, independientemente de que haya ejemplos de distintas clases en dicho nodo.

Los atributos introducidos al sistema deben ser simbólicos, al igual que la clase.

5.4. C4.5 en WEKA (J48)

La clase en la que se implementa el algoritmo C4.5 en la herramienta WEKA es *weka.classifers.j48.J48.java*. Las opciones que permite este algoritmo son las que se muestran en la tabla 2.3.

Tabla 5.2: *Opciones de configuración para el algoritmo C4.5 en WEKA.*

Opción	*Descripción*
minNumObj (2)	Número mínimo de instancias por hoja.

saveInstanceData (False)	Una vez finalizada la creación del árbol de decisión se eliminan todas las instancias que se clasifican en cada nodo, que hasta el momento se mantenían almacenadas.
binarySplits (False)	Con los atributos nominales también no se divide (por defecto) cada nodo en dos ramas.
unpruned (False)	En caso de no activar la opción, se realiza la poda del árbol.
subtreeRaising (True)	Se permite realizar el podado con el proceso *subtree raising*.
confidenceFactor (0.25)	Factor de confianza para el podado del árbol.
reducedErrorPruning (False)	Si se activa esta opción, el proceso de podado no es el propio de C4.5, sino que el conjunto de ejemplos se divide en un subconjunto de entrenamiento y otro de test, de los cuales el último servirá para estimar el error para la poda.
numFolds (3)	Define el número de subconjuntos en que hay que dividir el conjunto de ejemplos para, el último de ellos, emplearlo como conjunto de test si se activa la opción *reducedErrorPruning*.
useLaplace (False)	Si se activa esta opción, cuando se intenta predecir la probabilidad de que una instancia pertenezca a una clase, se emplea el *suavizado de Laplace*.

El algoritmo J48 se ajusta al algoritmo C4.5 al que se le amplían funcionalidades tales como permitir la realización del proceso de podado mediante *reducedErrorPruning* o que las divisiones sean siempre binarias *binarySplits*. Algunas propiedades concretas de la implementación son las siguientes:

- En primer lugar, en cuanto a los tipos de atributos admitidos, estos pueden ser simbólicos y numéricos. Se permiten ejemplos con faltas en dichos atributos, tanto en el momento de entrenamiento como en la predicción de dicho ejemplo. En cuanto a la clase, ésta debe ser simbólica.

- Se permiten ejemplos con peso.

- El algoritmo no posibilita la generación de reglas de clasificación a partir del árbol de decisión.

- Para el tratamiento de los atributos numéricos el algoritmo prueba los puntos secuencialmente, con lo que emplea tres de las cuatro opciones que se comentaron anteriormente (ver figura 2.3). La cuarta opción, que consistía en unir intervalos adyacentes con la misma clase mayoritaria no se realiza.

- También respecto a los atributos numéricos, cuando se intenta dividir el rango actual en dos subrangos se ejecuta la ecuación 2.14.

$$DivisiónMínima = 0.1 \times \frac{n_{ic}}{nc} \qquad (2.14)$$

En esta ecuación n_{ic} es el número de ejemplos de entrenamiento con el atributo *i* conocido, y *nc* el número de clases. Además, si el resultado de la ecuación es menor que el número mínimo de ejemplos que debe clasificarse por cada nodo hijo, se iguala a éste número y si es mayor que *25*, se iguala a dicho número. Lo que indica este número es el número mínimo de ejemplos que debe haber por cada uno de los dos nodos hijos que resultarían de la división por el atributo numérico, con lo que no se considerarían divisiones que no cumplieran este dato.

- El cálculo de la entropía y de la ganancia de información se realiza con las ecuaciones 2.15, 2.16 y 2.17.

$$G(A_i) = \frac{n_{ic}}{n^2}(I - I(A_i)) \qquad (2.15)$$

$$I = n_{ic}\, log_2(n_{ic}) - \sum_{c=1}^{nc} n_c log_2(n_c) \qquad (2.16)$$

$$I(A_i) = \sum_{j=1}^{nv(A_i)} n_{ij} log_2(n_{ij}) - I_{ij} \; ; \; I_{ij} = -\sum_{k=1}^{nc} n_{ijk} log_2(n_{ijk}) \qquad (2.17)$$

En estas ecuaciones, n_{ic} es el número de ejemplos con el atributo *i* conocido, *n* el número total de ejemplos, n_c el número de ejemplos conocidos (el atributo *i*) con clase *c*, n_{ij} el número de ejemplos con valor *j* en el atributo *i* y n_{ijk} el número de atributos con valor *j* en el atributo *i* y con clase *k*.

- Además, la información de ruptura se expresa como se muestra en la ecuación 2.18.

$$I(DivisiónA_i) = \frac{-\left(\sum_{j=1}^{nv(A_i)} n_{ij} log_2(n_{ij})\right) - n_{ic} log_2(n_{ic}) + n\, log_2(n)}{n} \qquad (2.18)$$

En la ecuación 2.18, n_{ij} es el número de ejemplos con valor *j* en el atributo *i*, n_{ic} es el número de ejemplos con valor conocido en el atributo *i* y *n* es el número total de ejemplos.

- El *suavizado de Laplace* se emplea en el proceso de clasificación de un ejemplar. Para calcular la probabilidad de que un ejemplo pertenezca a una clase determinada en un nodo hoja se emplea la ecuación 2.19.

$$P(k \mid E) = \frac{n_k + 1}{n + C} \qquad (2.19)$$

En la ecuación 2.19, n_k es el número de ejemplos de la clase clasificados en el nodo hoja, n el número total de ejemplos clasificados en el nodo y C el número de clases para los que hay algún ejemplo clasificado en el nodo.

5.5. Árbol de Decisión de un solo nivel en WEKA

La clase en la que se implementa el algoritmo tocón de decisión en la herramienta WEKA es *weka.classifers.DecisionStump.java*. Así, en WEKA se llama a este algoritmo *tocón de decisión* [decisión stump]. No tiene opciones de configuración, pero la implementación es muy completa, dado que admite tanto atributos numéricos como simbólicos y clases de ambos tipos también. El árbol de decisión tendrá tres ramas: una de ellas será para el caso de que el atributo sea desconocido, y las otras dos serán para el caso de que el valor del atributo del ejemplo de test sea igual a un valor concreto del atributo o distinto a dicho valor, en caso de los atributos simbólicos, o que el valor del ejemplo de test sea mayor o menor a un determinado valor en el caso de atributos numéricos.

En el caso de los atributos simbólicos se considera cada valor posible del mismo y se calcula la ganancia de información con el atributo igual al valor, distinto al valor y valores perdidos del atributo. En el caso de atributos simbólicos se busca el mejor punto de ruptura, tal y como se vio en el sistema C4.5 (ver punto 2.2.2.2).

Deben tenerse en cuenta cuatro posibles casos al calcular la ganancia de información: que sea un atributo simbólico y la clase sea simbólica o que la clase sea numérica, o que sea un atributo numérico y la clase sea simbólica o que la clase sea numérica. A continuación se comenta cada caso por separado.

Atributo Simbólico y Clase Simbólica

Se toma cada vez un valor v_x del atributo simbólico A_i como base y se consideran únicamente tres posibles ramas en la construcción del árbol: que el atributo A_i sea igual a v_x, que el atributo A_i sea distinto a v_x o que el valor del atributo A_i sea desconocido. Con ello, se calcula la entropía del atributo tomando como base el valor escogido tal y como se muestra en la ecuación 2.20.

$$I(A_{iv_x}) = \frac{\sum_{j=1}^{3} n_{ij} log(n_{ij}) - I_{ij}}{n\ log(2)} \ ; \ I_{ij} = \sum_{k=1}^{nc} n_{ijk} log(n_{ijk}) \qquad (2.20)$$

En la ecuación 2.20 el valor de j en el sumatorio va desde *1* hasta *3* porque los valores del atributo se restringen a tres: igual a v_x, distinto a v_x o valor desconocido. En cuanto a los parámetros, n_{ij} es el número de ejemplos con valor j en el atributo i, n el número total de ejemplos y n_{ijk} el número de ejemplos con valor j en el atributo i y que pertenece a la clase k.

Atributo Numérico y Clase Simbólica

Se ordenan los ejemplos según el atributo A_i y se considera cada z_x, definido como el punto medio entre los valores v_x y v_{x+1}, del atributo como posible punto de corte. Se consideran entonces como posibles valores del atributo el rango menor o igual a z_x, mayor a z_x y valor desconocido. Se calcula la entropía (ecuación 2.20) del rango tomando como base esos tres posibles valores restringidos del atributo.

Atributo Simbólico y Clase Numérica

Se vuelve a tomar como base cada vez un valor del atributo, tal y como se hacía en el caso *Atributo Simbólico y Clase Simbólica*, pero en este caso se calcula la varianza de la clase para los valores del atributo mediante la ecuación 2.21.

$$Varianza(A_{iv_x}) = \sum_{j=1}^{3} \left(SS_j - \frac{S_j}{W_j} \right) \qquad (2.21)$$

En la ecuación 2.21, S_j es la suma de los valores de la clase de los ejemplos con valor j en el atributo i, SS_j es la suma de los valores de la clase al cuadrado y W_j es la suma de los pesos de los ejemplos (número de ejemplos si no se incluyen pesos) con valor j en el atributo.

Atributo Numérico y Clase Numérica

Se considera cada valor del atributo como punto de corte tal y como se hacía en el caso *Atributo Numérico y Clase Simbólica*. Posteriormente, se calcula la varianza tal y como se muestra en la ecuación 2.21.

En cualquiera de los cuatro casos que se han comentado, lo que se busca es el valor mínimo de la ecuación calculada, ya sea la entropía o la varianza. De esta forma se obtiene el atributo que será raíz del árbol de decisión y sus tres ramas. Lo único que se hará por último es construir dicho árbol: cada rama finaliza en un nodo *hoja* con el valor de la clase, que será la media o la moda de los ejemplos que se clasifican por ese camino, según se trate de una clase numérica o simbólica.

5.6. 1R en WEKA

La clase *weka.classifers.OneR.java* implementa el algoritmo 1R. La única opción configurable es la que se muestra en la tabla 2.4.

Tabla 5.3: *Opciones de configuración para el algoritmo 1R en WEKA.*

Opción	***Descripción***
minBucketSize	Número mínimo de ejemplos que deben pertenecer a un

(6)	conjunto en caso de atributo numérico.

La implementación que se lleva a cabo en WEKA de este algoritmo cumple exactamente con lo descrito anteriormente.

Como vemos, 1R es un clasificador muy sencillo, que únicamente utiliza un atributo para la clasificación. Sin embargo, aún hay otro clasificador más sencillo, el 0R, implementado en *weka.classifers.ZeroR.java*, que simplemente calcula la media en el caso de tener una clase numérica o la moda, en caso de una clase simbólica. No tiene ningún tipo de opción de configuración.

5.7. PRISM en WEKA

La clase *weka.classifers.Prism.java* implementa el algoritmo PRISM. No tiene ningún tipo de configuración posible. Únicamente permite atributos nominales, la clase debe ser también nominal y no puede haber atributos con valores desconocidos. La implementación de esta clase sigue completamente el algoritmo expuesto en la figura 2.10.

5.8. PART en WEKA

La clase *weka.classifers.j48.PART.java* implementa el algoritmo PART. En la tabla 2.5 se muestran las opciones de configuración de dicho algoritmo.

Tabla 5.4: *Opciones de configuración para el algoritmo PART en WEKA.*

Opción	***Descripción***
minNumObj (2)	Número mínimo de instancias por hoja.
binarySplits (False)	Con los atributos nominales también no se divide (por defecto) cada nodo en dos ramas.
confidenceFactor (0.25)	Factor de confianza para el podado del árbol.
reducedErrorPruning (False)	Si se activa esta opción, el proceso de podado no es el propio de C4.5, sino que el conjunto de ejemplos se divide en un subconjunto de entrenamiento y otro de test, de los cuales el último servirá para estimar el error para la poda.
numFolds (3)	Define el número de subconjuntos en que hay que dividir el conjunto de ejemplos para, el último de ellos, emplearlo como conjunto de test si se activa la opción *reducedErrorPruning*.

Como se ve en la tabla 2.5, las opciones del algoritmo PART son un subconjunto de las ofrecidas por J48, que implementa el sistema C4.5, y es

que PART emplea muchas de las clases que implementan C4.5, con lo que los cálculos de la entropía, del error esperado,... son los mismos.

La implementación que se realiza en WEKA del sistema PART se corresponde exactamente con lo comentado anteriormente, y más teniendo en cuenta que los implementadores de la versión son los propios creadores del algoritmo.

Por último, en cuanto a los tipos de datos admitidos por el algoritmo, estos son numéricos y simbólicos para los atributos y simbólico para la clase.

5.9. *Naive* Bayesiano en WEKA

El algoritmo *naive* Bayesiano se encuentra implementado en la clase *weka.classifiers.NaiveBayesSimple.java*. No dispone de ninguna opción de configuración. El algoritmo que implementa esta clase se corresponde completamente con el expuesto anteriormente. En este caso no se usa el *estimador de Laplace*, sino que la aplicación muestra un error si hay menos de dos ejemplos de entrenamiento para una terna *atributo-valor-clase* o si la desviación típica de un atributo numérico es igual a *0*.

Una alternativa a esta clase que también implementa un clasificador *naive* Bayesiano es la clase *weka.classifiers.NaiveBayes.java*. Las opciones de configuración de que disponen son las mostradas en la tabla 2.6.

Tabla 5.5: *Opciones de configuración para el algoritmo Bayes naive en WEKA.*

Opción	***Descripción***
useKernelEstimator (False)	Emplear un estimador de densidad de núcleo (ver punto 2.3.3) para modelar los atributos numéricos en lugar de una distribución normal.

En este caso, sin embargo, en lugar de emplear la frecuencia de aparición como base para obtener las probabilidades se emplean distribuciones de probabilidad. Para los atributos discretos o simbólicos se emplean estimadores discretos, mientras que para los atributos numéricos se emplean bien un estimador basado en la distribución normal o bien un estimador de densidad de núcleo.

Se creará una distribución para cada clase, y una distribución para cada *atributo-clase*, que será discreta en el caso de que el atributo sea discreto. El estimador se basará en una distribución normal o *kernel* en el caso de los *atributo-clase* con atributo numérico según se active o no la opción mostrada en la tabla 2.6.

En el caso de los atributos numéricos, en primer lugar se obtiene la precisión de los rangos, que por defecto en la implementación será de *0,01* pero que se

calculará siguiendo el algoritmo descrito, mediante pseudocódigo, en la figura 2.15.

```
Precisión (ejemplos, atributo) {
  p = 0.01 // valor por defecto
  // se ordenan los ejemplos de acuerdo al atributo numérico
  Ordenar_ejemplos (ejemplos, atributo)
  vUltimo = Valor(ejemplos(0), atributo)
  delta = 0;
  distintos = 0;
  Para cada ejemplo (ej) de ejemplos
    vActual = Valor (ej, atributo)
    Si vActual <> vUltimo Entonces
      delta = delta + (vActual - vUltimo)
      vActual = vUltimo
      distintos = distintos + 1
  Si distintos > 0 Entonces
    p = delta / distintos
  Devolver p
}
```

Figura 5.1: *Algoritmo empleado para definir la precisión de los rangos para un atributo.*

Una vez obtenida la precisión de los rangos, se crea el estimador basado en la distribución correspondiente y con la precisión calculada. Se recorrerán los ejemplos de entrenamiento y de esta forma se generará la distribución de cada *atributo-clase* y de cada clase.

Cuando se desee clasificar un ejemplo el proceso será el mismo que se comentó anteriormente, y que se basaba en la ecuación 2.27, pero obteniendo las probabilidades a partir de estas distribuciones generadas. En el caso de los atributos numéricos, se calculará la probabilidad del rango *[x-precisión, x+precisión]*, siendo *x* el valor del atributo.

5.10. VFI en WEKA

El clasificador VFI se implementa en la clase *weka.classifiers.VFI.java*. Las opciones de configuración de que dispone son las que se muestran en la tabla 2.7.

Tabla 5.6: *Opciones de configuración para el algoritmo Bayes naive en WEKA.*

Opción	*Descripción*
weightByConfidence (True)	Si se mantiene activa esta opción cada atributo se pesará conforme a la ecuación 2.29.
bias (0.6)	Parámetro de configuración para el pesado por confianza.

El algoritmo que se implementa en la clase VFI es similar al mostrado en la figura 2.16. Sin embargo, sufre cambios sobretodo en el proceso de clasificación de un nuevo ejemplar:

- La normalización de los intervalos por clase se realiza durante la clasificación y no durante el entrenamiento.

- Si se activa la opción de *pesado por confianza*, cada voto de cada atributo a cada clase se pesa mediante la ecuación 2.29.

$$w(A_i) = I(A_i)^{bias} = \left(\frac{-\left(\sum_{i=0}^{nC} n_i \; lg(n_i)\right) + n \; lg(n)}{n \; lg(2)} \right)^{bias} \quad (2.29)$$

  En la ecuación 2.29 $I(A_i)$ es la entropía del atributo A_i, siendo n el número total de ejemplares, nC el número de clases y n_i el número de ejemplares de la clase i. El parámetro *bias* es el que se configuró como entrada al sistema, tal y como se mostraba en la tabla 2.7.

- En cuanto a los atributos, pueden ser numéricos y simbólicos, mientras que la clase debe ser simbólica.

Relacionado con este clasificador se encuentra otro que se implementa en la herramienta WEKA. Se trata de la clase *weka.classifiers.HyperPipes.java*. Este clasificador no tiene ningún parámetro de configuración y es una simplificación del algoritmo VFI: En este caso se almacena para cada atributo numérico el mínimo y el máximo valor que dicho atributo obtiene para cada clase, mientras que en el caso de los atributos simbólicos marca los valores que el atributo tiene para cada clase. A la hora de clasificar un nuevo ejemplo, simplemente cuenta, para cada clase, el número de atributos que se encuentran en el intervalo almacenado en el caso de atributos numéricos y el número de atributos simbólicos con valor activado en dicha clase. La clase con mayor número de coincidencias *gana*.

5.11. KNN en WEKA (IBk)

En WEKA se implementa el clasificador KNN con el nombre IBk, concretamente en la clase *weka.classifiers.IBk.java*. Además, en la clase *weka.classifiers.IB1.java* hay una versión simplificada del mismo, concretamente un clasificador NN [Nearest Neighbor], sin ningún tipo de opción, en el que, como su propio nombre indica, tiene en cuenta únicamente el voto del vecino más cercano. Por ello, en la tabla 2.8 se muestran las opciones que se permiten con el clasificador IBk.

Tabla 5.7: *Opciones de configuración para el algoritmo IBk en WEKA.*

Opción	***Descripción***
KNN (1)	Número de vecinos más cercanos.
distanceWeighting (No distance weighting)	Los valores posibles son: *No distance weighting*, *Weight by 1-distance* y *Weight by 1/distance*. Permite definir si se deben "pesar" los vecinos a la hora de votar bien según su semejanza o con la inversa de su distancia con respecto al ejemplo a clasificar.

crossValidate (False)	Si se activa esta opción, cuando se vaya a clasificar una instancia se selecciona el número de vecinos (hasta el número especificado en la opción KNN) mediante el proceso *hold-one-out*.
meanSquared (False)	Minimiza el error cuadrático en lugar del error absoluto para el caso de clases numéricas cuando se activa la opción *crossValidate*.
windowSize (0)	Si es *0* el número de ejemplos de entrenamiento es ilimitado. Si es mayor que *0*, únicamente se almacenan los *n* últimos ejemplos de entrenamiento, siendo *n* el número que se ha especificado.
debug (False)	Muestra el proceso de construcción del clasificador.
noNormalization (False)	No normaliza los atributos.

El algoritmo implementado en la herramienta WEKA consiste en crear el clasificador a partir de los ejemplos de entrenamiento, simplemente almacenando todas las instancias disponibles (a menos que se restrinja con la opción *windowSize*). Posteriormente, se clasificarán los ejemplos de test a partir del clasificador generado, bien con el número de vecinos especificados o comprobando el mejor *k* si se activa la opción *crossValidate*. En cuanto a los tipos de datos permitidos y las propiedades de la implementación, estos son:

- Admite atributos numéricos y simbólicos.
- Admite clase numérica y simbólica. Si la clase es numérica se calculará la media de los valores de la clase para los *k* vecinos más cercanos.
- Permite dar peso a cada ejemplo.
- El proceso de *hold-one-out* consiste en, para cada *k* entre *1* y el valor configurado en KNN (ver tabla 2.8), calcular el error en la clasificación de los ejemplos de entrenamiento. Se escoge el *k* con un menor error obtenido. El error cometido para cada *k* se calcula como el error medio absoluto o el cuadrático (ver tabla 2.8) si se trata de una clase numérica. El cálculo de estos dos errores se puede ver en las ecuaciones 2.33 y 2.34 respectivamente. Si la clase es simbólica se tomará como error el número de ejemplos fallados entre el número total de ejemplos.

$$MAE = \frac{\sum_{i=1}^{m} \left| y_i - \hat{y}_i \right|}{m} \qquad (2.33)$$

$$MSE = \frac{\sum_{i=1}^{m} \left(y_i - \hat{y}_i \right)^2}{m} \qquad (2.34)$$

En las ecuaciones 2.33 y 2.34 y_i es el valor de la clase para el ejemplo *i* e $\hat{y}_i$ es el valor predicho por el modelo para el ejemplo *i*. El número *m* será el número de ejemplos.

5.12. K* en WEKA

La clase en la que se implementa el algoritmo K* en la herramienta WEKA es *weka.classifers.kstar.KStar.java*. Las opciones que permite este algoritmo son las que se muestran en la tabla 2.9.

Tabla 5.8: *Opciones de configuración para el algoritmo K* en WEKA.*

Opción	***Descripción***
entropicAutoBlend (False)	Si se activa esta opción se calcula el valor de los parámetros x_0 (o s) basándose en la entropía en lugar del parámetro de mezclado.
globalBlend (20)	Parámetro de mezclado, expresado en tanto por ciento.
missingMode (Average column entropy curves)	Define cómo se tratan los valores desconocidos en los ejemplos de entrenamiento: las opciones posibles son *Ignore the Instance with missing value* (no se tienen en cuenta los ejemplos con atributos desconocidos), *Treat missing value as maximally different* (diferencia igual al del vecino más lejano considerado), *Normilize over the attributes* (se ignora el atributo desconocido) y *Average column entropy curves* (ver ecuación 2.41).

Dado que los autores de la implementación de este algoritmo en WEKA son los autores del propio algoritmo, dicha implementación se corresponde perfectamente con lo visto anteriormente. Simplemente son destacables los siguientes puntos:

- Admite atributos numéricos y simbólicos, así como pesos por cada instancia.

- Permite que la clase sea simbólica o numérica. En el caso de que se trate de una clase numérica se empleará la ecuación 2.45 para predecir el valor de un ejemplo de *test*.

$$v(a) = \frac{\sum_{i=1}^{n} P^*(b|a) * v(b)}{\sum_{i=1}^{n} P^*(b|a)} \qquad (2.45)$$

  En la ecuación 2.45 *v(i)* es el valor (numérico) de la clase para el ejemplo *i*, *n* el número de ejemplos de entrenamiento, y *P*(i|j)* la probabilidad de transformación del ejemplo *j* en el ejemplo *i*.

- Proporciona cuatro modos de actuación frente a pérdidas en los atributos en ejemplos de entrenamiento.

- Para el cálculo de los parámetros x_0 y s permite basarse en el parámetro *b* o en el cálculo de la entropía.

- Las ecuaciones para el cálculos de *P** y de la *esfera de influencia* no son las comentadas en la explicación del algoritmo, sino las empleadas en los ejemplos de las figuras 2.20 y 2.21.

5.13. Redes de Neuronas en WEKA

La clase en la que se implementan las redes de neuronas en weka es *weka.classifiers.neural.NeuralNetwork.java*. Las opciones que permite configurar son las que se muestran en la tabla 2.10.

Tabla 5.9: *Opciones de configuración para las redes de neuronas en WEKA.*

Opción	Descripción
momentum (0.2)	Factor que se utiliza en el proceso de actualización de los pesos. Se multiplica este parámetro por el peso en el momento actual (el que se va a actualizar) y se suma al peso actualizado.
validationSetSize (0)	Determina el porcentaje de patrones que se emplearán como test del sistema. De esta forma, tras cada entrenamiento se validará el sistema, y terminará el proceso de entrenamiento si la validación da un valor menor o igual a *0*, o si se superó el número de entrenamientos configurado.
nominalToBinaryFilter (False)	Transforma los atributos nominales en binarios.
learningRate (0.3)	Razón de aprendizaje. Tiene valores entre *0* y *1*.
hiddenLayers (a)	Determina el número de neuronas ocultas. Sus posibles valores son: *'a'=(atribs+clases)/2*, *'i'=atribs*, *'o'=clases*, *'t'=atribs+clases*.
validationThreshold (20)	Si el proceso de validación arroja unos resultados en cuanto al error que empeoran durante el *n* veces consecutivas (siendo *n* el valor de esta variable), se detiene el aprendizaje.
reset (True)	Permite al sistema modificar la razón de aprendizaje automáticamente (la divide entre *2*) y comenzar de nuevo el proceso de aprendizaje si el proceso de entrenamiento no converge.
GUI (False)	Visualización de la red de neuronas. Si se activa esta opción se puede modificar la red de neuronas, parar el proceso de entrenamiento en cualquier momento, modificar parámetros como el de la razón de aprendizaje,...
autoBuild (True)	El sistema construye automáticamente la red basándose en las entradas, salidas y el parámetro *hiddenLayers*.

normalizeNumericClass (True)	Normaliza los posibles valores de la clase si ésta es numérica, de forma que estén entre *–1* y *1*.
decay (False)	La razón de ganancia se modifica con el ciclo de aprendizaje: $\alpha = \alpha/n$, donde *n* es el número de ciclo de aprendizaje actual.
trainingTime (500)	Número total de ciclos de aprendizaje.
normalizeAttributes (True)	Normaliza los atributos numéricos para que estén entre *–1* y *1*.
randomSeed (0)	Semilla para generar los números aleatorios que inicializarán los parámetros de la red.

La implementación de redes de neuronas que se realiza en la herramienta se ciñe al algoritmo de retropropagación.

Algunas características que se pueden destacar de esta implementación son:

- Se admiten atributos numéricos y simbólicos.
- Se admiten clases numéricas (predicción) y simbólicas (clasificación).
- Permite la generación manual de redes que no se ciñan a la arquitectura mostrada anteriormente, por ejemplo, eliminando conexiones de neuronas de una capa con la siguiente.
- Como función sigmoidal se utiliza la restringida entre *0* y *1* (ver ecuación 2.48).
- Los ejemplos admiten pesos: Cuando se aprende con dicho ejemplo se multiplica la razón de aprendizaje por el peso del ejemplo. Todo esto antes de dividir la razón de aprendizaje por el número de ciclo de aprendizaje si se activa *decay*.

5.14. Regresión Lineal en WEKA

Es en la clase *weka.classifers.LinearRegression.java* en la que se implementa la regresión lineal múltiple. Las opciones que permite este algoritmo son las que se muestran en la tabla 2.11.

Tabla 5.10: *Opciones de configuración para el algoritmo de regresión lineal en WEKA.*

Opción	***Descripción***
AttributeSeleccionMethod (M5 method)	Método de selección del atributo a eliminar de la regresión. Las opciones son *M5 Method*, *Greedy* y *None*.
debug (False)	Muestra el proceso de construcción del clasificador.

La regresión lineal se construye tal y como se comentó anteriormente. Algunas propiedades de la implementación son:

- Admite atributos numéricos y nominales. Los nominales con *k* valores se convierten en *k-1* atributos binarios.
- La clase debe ser numérica.
- Se permite pesar cada ejemplo.

En cuanto al proceso en sí, si bien se construye la regresión como se comentó anteriormente, se sigue un proceso más complicado para eliminar los atributos.

El algoritmo completo sería el siguiente:

1. Construir regresión para los atributos seleccionados (en principio todos).

2. Comprobar la ecuación 2.64 sobre todos los atributos.

$$c_i = \left| \frac{b_i S_i}{S_c} \right| \qquad (2.64)$$

   En la ecuación 2.64, S_c es la desviación típica de la clase. Se elimina de la regresión el atributo con mayor valor si cumple la condición $c_i > 1.5$. Si se eliminó alguno, volver a *1*.

3. Calcular el error cuadrático medio (ecuación 2.63) y el factor *Akaike* tal y como se define en la ecuación 2.65.

$$AIC = (m - p) + 2p \qquad (2.65)$$

   En la ecuación 2.65 *m* es el número de ejemplos de entrenamiento, *p* el número de atributos que forman parte de la regresión al llegar a este punto.

4. Escoger un atributo:

   a. Si el método es *Greedy*, se generan regresiones lineales en las que se elimina un atributo distinto en cada una de ellas, y se escoge la regresión con menor error medio absoluto.

   b. Si el método es *M5*, se calcula el valor de c_i (ecuación 2.64) para todos los atributos y se escoge el menor. Se genera la regresión sin el atributo *i* y se calcula la regresión lineal sin dicho atributo. Se calcula el error medio absoluto de la nueva regresión lineal.

   c. Si el método es *None*, se finaliza el proceso.

5. Mejorar regresión. Se calcula el nuevo factor *Akaike* con la nueva regresión como es muestra en la ecuación 2.66.

$$AIC = \frac{MSE_c}{MSE}(m - p_c) + 2p \qquad (2.66)$$

En la ecuación 2.66 MSE_c es el error cuadrático medio absoluto de la nueva regresión lineal y p_c el número de atributos de la misma. Mientras, *MSE* es el valor obtenido en el punto *3* y *p* el número de parámetros al llegar al mismo. Si el valor nuevo de *AIC* es menor que el anterior, se actualiza éste como nuevo y se mantiene la nueva regresión lineal, volviendo a intentar mejorarla (volver a *4*). Si no es así, se finaliza el proceso.

5.15. Regresión Lineal Ponderada Localmente en WEKA

El algoritmo se implementa en la clase *weka.classifers.LWR.java*. Las opciones que permite configurar son las que se muestran en la tabla 2.12.

Tabla 5.11: *Opciones de configuración para el algoritmo LWR en WEKA.*

Opción	***Descripción***
weightingKernel (0)	Indica cuál va a ser el método para ponderar a los ejemplos de entrenamiento: 0, lineal; 1, inverso; 2, gaussiano.
debug (False)	Muestra el proceso de construcción del clasificador y validación de los ejemplos de test.
KNN (5)	Número de vecinos que se tendrán en cuenta para ser ponderados y calcular la regresión lineal. Si bien el valor por defecto es 5, si no se modifica o confirma se utilizan todos los vecinos.

En primer lugar, las ecuaciones que se emplean en los métodos para ponderar a los ejemplos de entrenamiento son: para el método inverso, la ecuación 2.67; para el método lineal, la ecuación 2.68; y para el método gaussiano, la ecuación 2.69.

$$\omega_i = max(1.0001 - d_{ij}, 0) \qquad (2.68)$$

$$\omega_i = e^{-d_{ij} * d_{ij}} \qquad (2.69)$$

El proceso que sigue el algoritmo es el que se comentó anteriormente. Algunas propiedades que hay que mencionar sobre la implementación son:

- Se admiten atributos simbólicos y numéricos.
- Se admiten ejemplos ya pesados, en cuyo caso, el peso obtenido del proceso explicado anteriormente se multiplica por el peso del ejemplo.

- Se toma como parámetro de suavizado la siguiente distancia mayor al del *k*-ésimo ejemplo más próximo.

- Para la generación de la regresión lineal se emplea la clase explicada en el punto anterior (ver punto 2.3.1.1), con los parámetros por defecto y con los ejemplos pesados.

5.16. M5 en WEKA

La clase en la que se implementa el algoritmo M5 en la herramienta WEKA es *weka.classifers.m5.M5Prime.java*. Las opciones que permite este algoritmo son las que se muestran en la tabla 2.13.

Tabla 5.12: *Opciones de configuración para el algoritmo M5 en WEKA.*

Opción	***Descripción***
ModelType (ModelTree)	Permite seleccionar como modelo a construir entre un árbol de modelos, un árbol de regresión o una regresión lineal.
useUnsmoothed (False)	Indica si se realizará el proceso de suavizado (*False*) o si no se realizará (*True*).
pruningFactor (2.0)	Permite definir el factor de poda.
verbosity (0)	Sus posibles valores son *0*, *1* y *2*, y permite definir las estadísticas que se mostrarán con el modelo.

En cuanto a la implementación concreta que se lleva a cabo en esta herramienta del algoritmo M5, cabe destacar lo siguiente:

- Admite atributos simbólicos y numéricos; la clase debe ser, por supuesto, numérica.

- Para la generación de las regresiones lineales se emplea la clase que implementa la regresión lineal múltiple en WEKA (punto 2.3.1.1).

- El número mínimo de ejemplos que deben clasificarse a través de un nodo para seguir dividiendo dicho nodo, definido en la constante *SPLIT_NUM* es *3.5*, mientras la otra condición de parada, que es la desviación típica de las clases en el nodo respecto a la desviación típica de todas las clases del conjunto de entrenamiento, está fijada en *0.05*.

- En realidad no se intenta minimizar el *SDR* tal y como se definió en la ecuación 2.71, sino que se intenta minimizar la ecuación 2.75, que se muestra a continuación.

$$SDR = \sqrt[5]{S^2} - \frac{n_I}{n}\sqrt[5]{S_I^2} - \frac{n_D}{n}\sqrt[5]{S_D^2} \qquad (2.75)$$

En la ecuación 2.75 n es el número total de ejemplos, n_I y n_D el número de ejemplos del grupo izquierdo y derecho respectivamente; y S, $S^2{}_I$ y $S^2{}_D$ la varianza del conjunto completo, del grupo izquierdo y del grupo derecho respectivamente, definiéndose la varianza como se muestra en la ecuación 2.76.

$$S^2 = \left| \frac{\sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}}{n} \right| \qquad (2.76)$$

En la ecuación 2.76 n es el número de ejemplos y x_i el valor de la clase para el ejemplo i.

- El cálculo del error de estimación para un nodo determinado, mostrado en la ecuación 2.73, se modifica ligeramente hasta llegar al que se muestra en la ecuación 2.77.

$$e(I) = \frac{n + pv}{n - v} \times \sqrt{\frac{\left| \sum_{i \in I} (y_i - \hat{y}_i)^2 - \left(\sum_{i \in I} y_i - \hat{y}_i) \right)^2 \right|}{n}} \qquad (2.77)$$

En la ecuación 2.77 p es el factor de podado que es configurable y, como se veía en la tabla 2.13, por defecto es *2*.

- Por último, la constante k empleada en el modelo de suavizado (ecuación 2.70) se configura con el valor *15*.

Por lo demás la implementación que se lleva a cabo respeta en todo momento el algoritmo mostrado en la figura 2.26.

5.17. Kernel Density en WEKA

Es en la clase *weka.classifiers.KernelDensity* en la que se implementa eel algoritmo de densidad de núcleo. No se puede configurar dicho algoritmo con ninguna propiedad. Además, sólo se admiten clases simbólicas, a pesar de que los algoritmos de densidad de núcleo, como se comentó anteriormente nacen como un método de estimación no paramétrica (clases numéricas). A continuación se muestran las principales propiedades de la implementación así como los atributos y clases permitidas:

- En cuanto a los atributos, pueden ser numéricos y simbólicos.
- La clase debe ser simbólica.
- Como función núcleo [kernel] se emplea la distribución normal o gaussiana (ecuación 2.83) normalizada, esto es, con media *0* y desviación típica *1*.
- Como tamaño de ventana se emplea $h = 1/\sqrt{n}$, siendo *n* el número de ejemplos de entrenamiento.
- Para clasificar el ejemplo A_i, para cada ejemplo de entrenamiento A_j se calcula la ecuación 2.92.

$$V(A_i, A_j) = K(dist(A_i, A_j) \times \sqrt{n}) \times \sqrt{n} \qquad (2.92)$$

En la ecuación 2.92, *dist* es la distancia entre el ejemplo de test y uno de los ejemplos de entrenamiento, definida tal y como se describe en la figura 2.19. El resultado de esta ecuación para el par A_i-A_j se sumará al resto de resultados obtenidos para la clase a la que pertenezca el ejemplo A_j.

El pseudocódigo del algoritmo implementado por WEKA es el que se muestra en la figura 2.30.

```
Kernel Density (ejemplo) {
  Para cada ejemplo de entrenamiento (E) Hacer
    prob = 1
    c = clase de E
    Para cada atributo de E (A) Hacer
      temp = V(ejemplo, A)
      Si temp < LB Entonces
        prob = prob * LB
      Si no
        prob = prob * temp
    probs[c] = probs[c] + prob
  Normalizar(probs)
}
```

Figura 5.2: *Pseudocódigo del algoritmo Kernel Density.*

La clase que obtenga una mayor probabilidad será la que resulte ganadora, y la que se asignará al ejemplo de test. En cuanto a la constante *LB*, se define en la ecuación 2.93.

$$LB = min^{1/t-1} \qquad (2.93)$$

En la ecuación 2.93 *min* es el número mínimo almacenable por un *double* en Java y *t* el número de atributos de los ejemplos (incluida la clase).

5.18. *k*-means en WEKA

El algoritmo de *k*-medias se encuentra implementado en la clase *weka.clusterers.SimpleKMeans.java*. Las opciones de configuración de que disponen son las que vemos en la tabla 2.14.

Tabla 5.13: *Opciones de configuración para el algoritmo k-medias en WEKA.*

Opción	***Descripción***
numClusters (2)	Número de clusters.
seed (10)	Semilla a partir de la cuál se genera el número aleatorio para inicializar los centros de los clusters.

El algoritmo es exactamente el mismo que el descrito anteriormente. A continuación se enumeran los tipos de datos que admite y las propiedades de la implementación:

- Admite atributos simbólicos y numéricos.
- Para obtener los centroides iniciales se emplea un número aleatorio obtenido a partir de la semilla empleada. Los *k* ejemplos correspondientes a los *k* números enteros siguientes al número aleatorio obtenido serán los que conformen dichos centroides.
- En cuanto a la medida de similaridad, se emplea el mismo algoritmo que el que veíamos en el algoritmo KNN (figura 2.19).
- No se estandarizan los argumentos, sino que se normalizan (ecuación 2.96).

$$\frac{x_{il} - min_l}{Max_l - min_l} \qquad (2.96)$$

En la ecuación 2.96, x_{if} será el valor *i* del atributo *f*, siendo min_f el mínimo valor del atributo *f* y Max_f el máximo.

5.19. COBWEB en WEKA

El algoritmo de COBWEB se encuentra implementado en la clase *weka.clusterers.Cobweb.java*. Las opciones de configuración de que disponen son las que vemos en la tabla 2.15.

Tabla 5.14: *Opciones de configuración para el algoritmo COBWEB en WEKA.*

Opción	***Descripción***

acuity (100)	Indica la mínima varianza permitida en un cluster
cutoff (0)	Factor de poda. Indica la mejora en utilidad mínima por una subdivisión para que se permita llevar a cabo.

La implementación de COBWEB en WEKA es similar al algoritmo explicado anteriormente. Algunas características de esta implementación son:

- Se permiten atributos numéricos y simbólicos.
- La semilla para obtener números aleatorios es fija e igual a *42*.
- Permite pesos asociados a cada ejemplo.
- Realmente el valor de cutoff es $0.01 \times 1 / \left(2\sqrt{\pi}\right)$.
- En el caso de que el ejemplo que se desea clasificar genere, en un nodo determinado, un *CU* menor al *cutoff*, se eliminan los hijos del nodo (poda).

5.20. EM en WEKA

El algoritmo EM se encuentra implementado en la clase *weka.clusterers.EM.java*. Las opciones de configuración de que disponen son las que vemos en la tabla 2.16.

Tabla 5.15: *Opciones de configuración para el algoritmo EM en WEKA.*

Opción	Descripción
numClusters (-1)	Número de clusters. Si es número es *–1* el algoritmo determinará automáticamente el número de clusters.
maxIteration (100)	Número máximo de iteraciones del algoritmo si esto no convergió antes.
debug (False)	Muestra información sobre el proceso de clustering.
seed (100)	Semilla a partir de la cuál se generan los número aleatorios del algoritmo.
minStdDev ($1e^{-6}$)	Desviación típica mínima admisible en las distribuciones de densidad.

En primer lugar, si no se especifica el número de clusters, el algoritmo realiza un primer proceso consistente en obtener el número óptimo de clusters. Se realiza mediante validación cruzada con *10* conjuntos [folders]. Se va aumentando el número de clusters hasta que se aumenta y empeora el resultado. Se ejecuta el algoritmo en diez ocasiones, cada una de ellas con nueve conjuntos de entrenamiento, sobre los que se ejecuta EM con los parámetros escogidos y posteriormente se valida el sistema sobre el conjunto de test, obteniendo como medida la *verosimilitud* sobre dicho conjunto. Se calcula la media de las diez medidas obtenidas y se toma como base para determinar si se continúa o no aumentando el número de clusters.

Una vez seleccionado el número óptimo de clusters, se procede a ejecutar el algoritmo EM sobre el conjunto total de entrenamiento hasta un máximo de iteraciones que se configuró previamente (ver tabla 2.16) si es que el algoritmo no converge previamente.

En cuanto a los tipos de atributos con admite el algoritmo y algunas propiedades interesantes, éstas son:

- En cuanto a los atributos, éstos pueden ser numéricos o simbólicos.
- Se entiende que se converge si en la siguiente iteración la verosimilitud aumenta en menos de $1e^{-6}$.
- No tiene en cuenta posibles correlaciones entre atributos.

5.21. Asociación A Priori en WEKA

La clase en la que se implementa el algoritmo de asociación A Priori es *weka.associations.Apriori.java*. Las opciones que permite configurar son las que se muestran en la tabla 2.17.

Tabla 5.16: *Opciones de configuración para el algoritmo de asociación A Priori en WEKA.*

Opción	Descripción
numRules (10)	Número de reglas requerido.
metricType (Confidence)	Tipo de métrica por la que ordenar las reglas. Las opciones son Confidence (confianza, ecuación 2.106), Lift (ecuación 2.107), Leverage (ecuación 2.108) y Conviction (ecuación 2.109).
minMetric	Mínimo valor de la métrica empleada. Su valor por defecto depende del tipo de métrica empleada: *0.9* para Confidence, *1.1* para Lift y Conviction y *0.1* para Leverage.
delta (0.05)	Constante por la que va decreciendo el soporte en cada iteración del algoritmo.
upperBoundMinSupport (1.0)	Máximo valor del soporte de los *item-sets*. Si los item-sets tienen un soporte mayor, no se les toma en consideración.
lowerBoundMinSupport (0.1)	Mínimo valor del soporte de los *item-sets*.
significanceLevel (-1.0)	Si se emplea, las reglas se validan para comprobar si su correlación es estadísticamente significativa (del nivel requerido) mediante el test χ^2. En este caso, la métrica a utilizar es Confidence.
removeAllMissingsCols (False)	Si se activa, no se toman en consideración los atributos con todos los valores perdidos.
-I (sólo modo texto)	Si se activa, se muestran los *itemsets* encontrados.

En primer lugar, el algoritmo que implementa la herramienta WEKA es ligeramente distinto al explicado anteriormente. En la figura 2.36 se muestra el algoritmo concreto.

```
Apriori (ejemplos, MS, mS) { /* MS: Máx. soporte;  mS: Mín. soporte */
   S = MS-delta
   Mientras No Fin
      Generar ItemSets en rango (MS, S)
      GenerarReglas (ItemSets)
      MS = MS-delta
      S = S-delta
      Si suficientes reglas O S menor que mS Entonces
         Fin
}

GenerarReglas (ItemSets) {
   Para cada ItemSet
      Generar posibles reglas del ItemSet
      Eliminar reglas según la métrica
}
```

Figura 5.3: *Algoritmo A Priori en WEKA.*

Así, el algoritmo no obtiene de una vez todos los *item-sets* entre los valores máximo y mínimo permitido, sino que se va iterando y cada vez se obtienen los de un rango determinado, que será de tamaño *delta* (ver tabla 2.17).

Además, el algoritmo permite seleccionar las reglas atendiendo a diferentes métricas. Además de la confianza (ecuación 2.106), se puede optar por una de las siguientes tres métricas.

- **Lift:** Indica cuándo una regla es mejor prediciendo el resultado que asumiendo el resultado de forma aleatoria. Si el resultado es mayor que uno, la regla es buena, pero si es menor que uno, es peor que elegir un resultado aleatorio. Se muestra en la ecuación 2.107.

$$lift(A \Rightarrow B) = \frac{confianza(A \Rightarrow B)}{P(B)} \qquad (2.107)$$

- **Leverage:** Esta medida de una regla de asociación indica la proporción de ejemplos adicionales cubiertos por dicha regla (tanto por la parte izquierda como por la derecha) sobre los cubiertos por cada parte si fueran independientes. Se muestra en la ecuación 2.108.

$$leverage(A \Rightarrow B) = P(A \cap B) - P(A) * P(B) \qquad (2.108)$$

- **Convicción:** Es una medida de implicación. Es direccional y obtiene su máximo valor (infinito) si la implicación es perfecta, esto es, si siempre que *A* ocurre sucede también *B*. Se muestra en la ecuación 2.109.

$$convicción(A \Rightarrow B) = \frac{P(A) * P(!B)}{P(A \cap !B)} \qquad (2.109)$$

Por último, cabe destacar que esta implementación permite únicamente atributos simbólicos. Además, para mejorar la eficiencia del algoritmo en la búsqueda de *item-sets*, elimina todos los atributos que tengan sus valores desconocidos en todos los ejemplos.

Capítulo 6. Ejemplos sobre casos de estudio

Bibliografía

[ACO02] Acosta Franco, Javier. "*Aplicación de los Sistemas Clasificadores tradicionales al análisis de datos. Adquisición automática de reglas*". Proyecto Fin de Carrera, Universidad Carlos III de Madrid, 2002.

[AGR96] A. Agresti. *An Introduction to Categorical Data Analysis.* New York: John Wiley & Sons, 1996.

[AIS93b] R. Agrawal, T. Imielinski, and A. Swami. *Mining association rules between sets of items in large databases*. In Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'93), pages 207-216, Washington, DC, May 1993.

[AKA73] Akaike, A. (1973). "*Information theory and an extension of the maximum likelihood principle*" In Petrov, B. N., and Saki, F. L.(eds.), *Second International Symposium of Information Theory*. Budapest.

[AS94a] R. Agrawal and R. Srikant. "*Fast algorithms for mining association rules in large databases*". In Research Report RJ 9839, IBM Almaden Research Center, San Jose, CA, June 1994.

[AS94b] R. Agrawal and R. Srikant. "*Fast algorithms for mining association rules*". In Proc. 19941nt. Conf VeryLargeData Bases (VLDB'94), pages 487-499, Santiago, Chile, Sept.1994.

[BERR97] M.Berry, and G.Linoff, , "*Data Mining Techniques for Marketing, Sales, and Customer Support*" John Wiley, NY, 1997.

[BMS97] S. Brin, R. Motwani, and C. Silverstein. "*Beyond market basket: Generalizing association rules to correlations*". In *Proc. 1997ACM-SIGMOD I of Data (SIGMOD'97),* pages 265-276, Tucson, AZ, May 1997.

[BRIS96] G. Briscoe, and T. Caelli, "*A Compendium of Machine Learning*. *Vol. 1: Symbolic Machine Learning.*" Ablex Publishing Corporation, New Jersey, 1996.

[CEN87] J. Cendrowska (1987). "*PRISM: An algorithm for inducing modular rules*". International Journal of Man-Machine Studies. Vol.27, No.4, pp.349-370.

[CHSVZ] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, A.Zanasi, *Discovering Data Mining From concept to implementation*. Prentice Hall 1997.

[CHY96] M.S. Chen, J. Han, and P. S. Yu. "*Data mining: An overview from a database perspective*". IEEE Trans. Knowledge and Data Engineering, 8:866-883, 1996.

[CLTR95] John, G. Cleary and Leonard, E. Trigg (1995) "*K*: An Instance-based Learner Using an Entropic Distance Measure*", *Proceedings of the 12th International Conference on Machine learning*, pp. 108-114.

[CN89] P. Clark and T. Niblett. "*The CN2 induction algorithm. Machine Learning".* 3:261-283, 1989.

[CODD70] E. F. Codd, "*A Relational Model of Data for Large Shared Data Banks*," Communications of the ACM, Vol. 13, 1970.

[CR95] Y.Chauvin and D. Rumelhart. "*Backpropagation: Theory, Architectures, and Applications*". Hillsdale, NJ: Lawrence Erlbaurn Assoc., 1995.

[DARP98] *Workshop on Knowledge Discovery in Databases*, Defense Advanced Research Projects Agency, Pittsburgh, PA, June 1998.

[DEA97] Deaton, A. (1997): "*The Analysis of Household Surveys. A Microeconometric Approach to Development Policy. The World Bank*". The Johns Hopkins University Press.

[DECI] *Decision Support Journal*, Elsevier/North Holland Publications.

[DEGR86] T. DeGroot, "*Probability and Statistics*," Addison Wesley, MA, 1986.

[DEV95] J. L. Devore. "*Probability and Statistics for Engineering and the Sciences*". 4th ed. New York: Duxbury Press, 1995.

[DFL96] DiNardo, J., Fortin, N. and Lemieux, T. (1996): "*Labor Market Institutions and the Distribution of Wages, 1973-1992: a Semiparametric Approach. Econometrica*", Vol. 64, No.5. September.

[DH73] R. Duda and P. Hart. “*Pattern Classification and Scene Analysis*”. New York: John Wiley & Sons, 1973.

[DOB90] A. J. Dobson. “*An Introduction to Generalized Linear Models*”. New York: Chapman and Hall, 1990.

[FAYY96] U. Fayyad, et al. "*Advanced in Knowledge Discovery and Data Mining*," MIT Press, MA, 1996.

[FIS87] D. Fisher, “*Improving inference through conceptual clustering*”. In Proc. 1987 AAAI Conf., pages 461-465, Seattle, WA, July 1987.

[FRWI98] Eibe Frank and Ian H. Witten (1998). “*Generating Achúrate Rule Sets Without Global Optimization.*” In Shavlik, J., ed., *Machine Learning: Proceedings of the Fifteenth International Conference*, Morgan Kaufmann Publishers, San Francisco, CA.

[FU94] L. Fu, “*Neural Networks in Computer Intelligence*”, New York, McGraw Hill, 1994

[FUR87] Furnas, G. W. et al. “*The vocabulary problem in human system communication*”. Communications of the ACM, 30, nº 11, Nov. 1987.

[HALI94] Härdle, W. and Linton, O. (1994): “*Applied Nonparametric Methods. Handbook of Econometrics*”, Volume IV, Chapter 38. Elsevier Science.

[HH96] Hearst, M.; Hirsh, H. (eds.) Papers from the AAAI Spring Symposium on Machine Learning in information Access, Stanford, March 25-27, 1996. *http://www.parc.xerox.com/istl/projects/mlia*

[HMM86] J. Hong, I. Mozetic, and R. S. Michalski. “*AQ15: Incremental learning of attribute-based descriptions from examples, the method and user's guide*”. In Report ISG 85-5, UIUCDCS-F-86-949, Department of Computer Science, University of Illinois at Urbana-Champaign, 1986.

[HOL93] R.C. Holte (1993). “*Very simple classification rules perform well on most commonly used datasets*”. Machine Learning, Vol. 11, pp. 63-91.

[IEEE89] "*Parallel Architectures for Databases*," IEEE Tutorial, 1989 (Ed: A. Hurson et al.).

[JAM85] M. James. “*Classification Algorithms*”*.* New York: John Wiley & Sons, 1985.

[JOH97] G. H. John. “*Enhancements to the Data Mining Process*”. Ph.D. Thesis, Computer Science Dept., Stanford University, 1997.

[KB00] Kosala, R.; Blockeel, H. "*Web Mining Research: A Survey*" ACM SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining, June 2000, Vol. 2, nº 1, pp. 1-15

[KODR90] Kodratoff, Y. and Michalski, R.S., "*Machine Learning and Artificial Inteligence Approach, Vol. 3*", San Mateo, CA: Morgan Kaufmann, 1990

[LAN96] P. Langley. "*Elements of Machine Learning*". San Francisco: Morgan Kaufmann, 1996.

[LOP01] A. López Cilleros, "*Modelización del Comportamiento Humano para un Agente de la Robocup mediante Aprendizaje Automático*". Proyecto Fin de Carrera, Universidad Carlos III de Madrid, 2001.

[MAC67] MacQueen. "*Some methods for classification and analysis of multivariate observations*". Proc. 5th Berkeley Symp. Math. Statisi. Prob., 1:281-297, 1967.

[MBK98] R. S. Michalski, I. Brakto, and M. Kubat. *Machine Learning and Data Mining. Methods and Applications.* New York: John Wiley & Sons, 1998.

[MIT97] T. Mitchell, "*Machine Learning*," McGraw Hill, NY, 1997.

[MM95] J. Major and J. Mangano. "*Selecting among rules induced from a hurricane database*. Journal of Intelligent Information Systems", 4:39-52, 1995.

[MORE98a] D. Morey, "*Knowledge Management Architecture*" Handbook of Data Management, Auerbach Publications, NY, 1998 (Ed: B. Thuraisingham).

[MS83] R.S. Michalski, and R.E. Stepp, "*Learning from observation: Conceptual clustering*". In R.S. Michalski, J.G. Carbonell, and Mitchell, T.M, editors, Machine Learning: An Artificial Intelligence Approach, Vol 1. San Mateo, CA: Morgan Kaufmann, 1983.

[PSF91] G. Piatesky-Shapiro and W.J. Frawley. *Knowledge Discovery in Databases*. Cambridge, MA:AAA/MIT Press, 1991

[PTVF96] W. H. Press, S. A. Teukolosky, W. T. Vetterling, and B. P Flannery. "*Numerical Recipes in C: The Art of Scientific Computing*". Cambridge, UK: Cambridge University Press, 1996.

[QUIN79] J.R.Quinlan, "*Discovering Rules from Large Collections of Examples*", In Expert Systems in the Microelectronic Age, Michie, D. (Ed.), Edimburgo University Press, Edimburgo. 1979

[QUIN86] J.R. Quinlan, "*Induction of Decision Trees (ID3 algorithm*)", Machine Learning J., vol. 1, núm. 1, pp. 81-106. 1986

[QUIN87] J.R. Quinlan, "Simplifying decision trees", International Journal of Man-Machine Studies, núm. 27, pp. 221-234. 1987

[QUIN93] J.R. Quinlan, "*C4.5: Programs for Machine Learnirig*," Morgan Kaufmann, CA, 1993.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "*Learning internal representations by error propagation*". In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing.* Cambridge, MA: MIT Press, 1986.

[RIO02] Ríos Montaño, Pablo Miguel. "*Sistemas Clasificadores Extendidos (XCS). Desarrollo de una librería y comparación CS vs. XCS*". Proyecto Fin de Carrera, Universidad Carlos III de Madrid, 2002.

[RM86] D. E. Rumelhart and J. L. McClelland. "*Parallel Distributed Processing*". Cambridge, MA: MIT Press, 1986.

[RMS98] S. Ramaswamy, S. Mahajan, and A. Silberschatz. "*On the discovery of interesting patterns in association rules*". In Proc. *1998* Int. Conf Very Large Data Bases (VLDB'98), pages 368-379, New York, Aug. 1998.

[SIL86] Silverman, B. W. (1986): "*Density Estimation for Statistics and Data Analysis*", London and New York, Chapman and Hall.

[SLK96] E. Simoudis, B. Livezey and R. Kerber. *Integrating Inductive and Deductive Reasoning for Data Mining*. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smiyth, and R. Uthurusamy, editors, Advances in knowledge Discovery and Data Mining, pages 353-373. Cambridge, MA:AAAI/MIT Press, 1996

[SN88] K. Saito and R. Nakano. "*Medical diagnostic expert system based on PDP model*". In Proc. IEEE International Conf on Neural Networks, volume 1, pages 225262. San Mateo, CA: 1988.

[THUR99] B. Thuraisingham, "*Data Mining: Technologies, Techniques, Tools and Trends*." CRC Press, 1999

[UTG88] P.E. Utgoff, "*An Incremental ID3*." In Proc. Fifth Int. Conf. Machine Learning, pages 107-120, San Mateo, CA, 1988

[VIS95] Proceedings of the 1995 *Workshop on Visualization and Databases*, Atlanta, GA, October 1997 (Ed: G. Grinstein)

[VIS97] Proceedings of the 1997 *Workshop on Visualization and Data Mining* , Phoenix, AZ, October 1997 (Ed: G. Grinstein).

[WF00] H. Witten, and E Frank (2000). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. San Francisco, CA: Morgan Kaufmann.

[WI98] S.M Weiss, and Indurkhya. "*Predictive Data Mining*". San Francisco: Morgan Kaufmann 1998

[WK91] S.M. Weiss and C. A. Kulikowski. "*Computer Systems That Learn: Classification and Prediction Methodsfrom Statistics, Neural Nets, Machine Learning, and Expert Systems*". San Mateo, CA: Morgan Kaufmann, 199 1.

[ACM90] Special Issue on Heterogeneous Database Systems, ACM Computing Surveys, September 1990.

[ACM91] Special Issue on Next Generation Database Systems, Communications of the ACM, October 1991.

[ACM95] Special Issue on Digital Libraries, Communications of the ACM, May 1995.

[ACM96a] Special Issue on Data Mining, Communications of the ACM, November 1996.

[ACM96b] Special Issue on Electronics Commerce, Communications of the ACM, June 1996.

[ADRI96] Adriaans, P., and Zantinge, D., "Data Mining," Addison Wesley, MA, 1996.

[AFCE97] Proceedings of the First Federal Data Mining Symposium, Washington D.C., December 1997.

[AFSB83] Air Force Summer Study Board Report on Multilevel Secure Database Systems, Department of Defense Document, 1983.

[AGRA93] Agrawal, A. et al.., "Database Mining a Performance Perspective," IEEE Transactions on Knowledge ani Data Engineering, Vol. 5, December 1993.

[BANE87] Banerjee, J. et al., "A Data Model for Object-Oriented Applications," ACM Transactions on Office Infonnation Systems, Vol. 5, 1987.

[BELL92] Bell D. and Grimson, J., "Distributed Database Systems," Addison Wesley, MA, 1992.

[BENS95] Bensley, E. et al., "Evolvable Systems Initiative for Realtime Command and Control Systems," Proceedings of the Ist IEEE Complex Systems Conference, Orlando, FL, November 1995.

[BERN87] Bernstein, P. et al., "Concurrency Control and Recovery in Database Systems," Addison Wesley, MA, 1987.

[BERR97] Berry, M. and Linoff, G., "Data Mining Techniques for Marketing, Sales, and Customer Support," John Wiley, NY, 1997.

[BRIS96] Briscoe, G., Caelli, T. "A Compendium of Machine Learning. Vol. 1: Symbolic Machine Learning*.”* Ablex Publishing Corporation, New Jersey, 1996.

[BROD84] Brodie, M. et al., "On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases, and Programming Languages," Springer Verlag, NY, 1984.

[BROD86] Brodie, M. and Mylopoulos, J., "On Knowledge Base Management Systems," Springer Verlag, NY, 1986.

[BROD88] Brodie, M. et al., "Readings in Artificial Intelligence and Databases," Morgan Kaufmann, CA, 1988.

[BROD95] Brodie M. and Stonebraker, M., "Migrating Legacy Databases," Morgan Kaufmann, CA, 1995.

[BUNE82] Buneman, P., "Functional Data Model," ACM Transactions on Database Systems, 1983.

[CARB98] Carbone, P., "Data Mining," Handbook of Data Management, Auerbach Publications, NY, 1998 (Ed: B. Thuraisingham).

[CERI84] Ceri, S. and Pelagatti, G., "Distributed Databases, Principles and Systems," McGraw Hill, NY, 1984.

[CHAN73] Chang C., and Lee R., "Symbolic Logic and Mechanical Theorem Proving," Academic Press, NY, 1973.

[CHEN76] Chen, P., "The Entity Relationship Model - Toward a Unified View of Data," ACM Transactions on Database Systems, Vol. 1, 1976.

[CHOR94] Chorafas, D., "Intelligent Multimedia Databases," Prentice Hall, NJ, 1994.

[CLIF96a] Clifton, C, and Morey, D., "Data Mining Technology Survey," Private Communication, Bedford, MA, December 1996.

[CLIF96b] Clifton, C. and Marks, D., "Security and Privacy Issues for Data Mining," Proceedings of the ACM SIGMOD Conference Workshop on Data Mining, Montreal, Canada, June 1996.

[CLIF98a] Clifton, C., "Image Mining," Private Communication, Bedford, MA, July 1998.

[CLIF98b] Clifton C., "Privacy Issues for Data Mining," Private Communication, Bedford, MA, April 1998.

[CODD70] Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, Vol. 13, 1970.

[COOL98] Cooley, R., "Taxonomy for Web Mining," Private Communication, Bedford, MA, August 1998.

[DARPA98] Workshop on Knowledge Discovery in Databases, Defense Advanced Research Projects Agency, Pittsburgh, PA, June 1998.

[DAS92] Das, S., "Deductive Databases and Logic Programming," Addison Wesley, MA, 1992.

[DATE90] Date, C. J., "An Introduction to Database Management Systems," Addison Wesley, MA, 1990 (6th edition published in 1995 by Addison Wesley).

[DCI96] Proceedings of the DCI Conference on Databases and Client Server Computing, Boston, MA, March 1996.

[DE98] Proceedings of the 1998 Data Engineering Conference, Orlando, FL, February 1998.

[DECI] Decision Support Journal, Elsevier/North Holland Publications.

[DEGR86] DeGroot, T., "Probability and Statistics," Addison Wesley, MA, 1986.

[DEVL88] Devlin, B. and Murphy, P.T., “An Architecture for a Bussiness and Information System”. *IBM Sys, J 27*, No 1, 1988

[DIGI95] Proceedings of the Advances in Digital Libraries Conference, McLean, VA, May 1995, (Ed: N. Adam et al.).

[DIST98] Workshop on Distributed and Parallel Data Mining, Melbourne, Australia, April 1998.

[DMH94] Data Management Handbook, Auerbach Publications, NY, 1994 (Ed: B. von Halle and D. Kull).

[DMH95] Data Management Handbook Supplement, Auerbach Publications, NY, 1995 (Ed: B. von Halle and D. Kull).

[DMH96] Data Management Handbook Supplement, Auerbach Publications, NY, 1996 (Ed: B. Thuraisingham).

[DMH98] Data Management Handbook Supplement, Auerbach Publications, NY, 1998 (Ed: B. Thuraisingharn).

[DOD94] Proceedings of the 1994 DoD Database Colloquium, San Diego, CA, August 1994.

[DOD95] Proceedings of the 1994 DoD Database Colloquium, San Diego, CA, August 1995.

[DSV98] DSV Laboratory, "Inductive Logic Programming," Private Communication, Stockholm, Sweden, June 1998.

[FAYY96] Fayyad, U. et al., "Advanced in Knowledge Discovery and Data Mining," MIT Press, MA, 1996.

[FELD95] Feldman, R. and Dagan, I., "Knowledge Discovery in Textual Databases (KDT)," Proceedings of the 1995 Knowledge Discovery in Databases Conference, Montreal, Canada, August 1995.

[FOWL97] Fowler, M. et al., "UML Distilled: Applying the Standard Object Modeling Language," Addison Wesley, MA, 1997.

[FROS86] Frost, R., "On Knowledge Base Management Systems," Collins Publishers, U.K., 1986.

[GALL78] Gallaire, H. and Minker, J., "Logic and Databases," Plenum Press, NY, 1978.

[GOLD89] Goldberg, D., “Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Weslwy, 1989

[GRIN95] Grinstein, G. and Thuraisingham, B., "Data Mining and Visualization: A Position Paper," Proceedings of the Workshop on Databases in Visualization, Atlanta GA, October 1995.

[GRUP98] Grupe F. and Owrang, M., "Database Mining Tools", in the Handbook of Data Management Supplement, Auerbach Publications, NY, 1998 (Ed: B.Thuraisingham).

[HAN98] Han, J., "Data Mining," Keynote Address, Second Pacific Asia Conference on Data Mining, Melbourne, Australia, April 1998.

[HAN98] Han, J. and Kamber, M., "Data Mining: Concepts and Techniques," ACADEMIC Press, 2001.

[HINK88] Hinke T., "Inference and Aggregation Detection in Database Management Systems," Proceedings of the 1988 Conference on Security and Privacy, Oakland, CA, April 1988.

[ICTA97] Panel on Web Mining, International Conference on Tools for Artificial Intelligence, Newport Beach, CA, November 1997.

[EEE89] "Parallel Architectures for Databases," IEEE Tutorial, 1989 (Ed: A. Hurson et al.).

[IEEE91] Special Issue in Multidatabase Systems, IEEE Computer, December 1991.

[IEEE98] IEEE Data Engineering Bulletin, June 1998.

[IFIP] Proceedings of the IDFIP Conference Series in Database Security, North Holland.

[IFIP97] "Web Mining," Proceedings of the 1997 IFIP Conference in Database Security, Lake Tahoe, CA, August 1997..

[ELP97] Summer School on Inductive Logic Programming, Prague, Czech Republic, September 1998.

[INMO88] Inmon, W., "Data Architecture: The Information Paradigm," Wellesley, Mas: QED Information Sciences, 1988.

[INMO93] Inmon, W., "Building the Data Warehouse," John Wiley and Sons, NY, 1993.

[JUNG98] Junglee Corporation, "Virtual Database Technology, XML, and the Evolution of the Web," IEEE Data Engineering Bulletin, June 1998 (authors: Prasad and Rajaraman).

[KDD95] Proceedings of the First Knowledge Discovery in Databases Conference, Montreal, Canada, August 1995.

[KDD96] Proceedings of the Second Knowledge Discovery in Databases Conference, Portland, OR, August 1996.

[KDD97] Proceedings of the Third Knowledge Discovery in Databases Conference, Newport Beach, CA, August 1997.

[KDD98] Proceedings of the Fourth Knowledge Discovery in Databases Conference, New York, NY, August 1998.

[KDP98] Panel on Privacy Issues for Data Mining, Knowledge Discovery in Databases Conference, New York, NY, August 1998.

[KDT98] Tutorial on Commercial Data Mining Tools, Knowledge Discovery in Databases Conference, August 1998 (Presenters: J. Elder and D. Abbott)

[KIM85] Kim, W. et al., "Query Processing in Database Systems," Springer Verlag, NY, 1985.

[KODR90] Kodratoff, Y. and Michalski, R.S., “Machine Learning and Artificial Inteligence Approach, Vol. 3, San Mateo, CA: Morgan Kaufmann, 1990

[KORT86] Korth, H. and Silberschatz, A., "Database System Concepts," McGraw Hill, NY, 1986.

[KOWA74] Kowalski, R. A., "Predicate Logic as a Programming Language," Information Processing 74, Stockholm, North Holland Publications, 1974.

[LIN97] Lin, T.Y., (Editor) "Rough Sets and Data Mining," Kluwer Publishers, MA, 1997.

[LLOY87] Lloyd, J., "Foundations of Logic Programming," Springer Verlag, Germany, 1987.

[LOOM95] Loomis, M., "Object Databases," Addison Wesley, MA, 1995.

[MAIE83] Maier, D., "Theory of Relational Databases," Computer Science Press, MD, 1983.

[MATTO98] Mattox, D. et al., "Software Agents for Data Management," Handbook of Data Management, Auerbach Publications, NY, 1998 (Ed: B. Thuraisingham).

[MDDS94] Proceedings of the Massive Digital Data Systems Workshop, published by the Community Management Staff, Washington D.C., 1994.

[MERL97] Merlino, A. et al., "Broadcast News Navigation using Story Segments," Proceedings of the 1997 ACM Multimedia Conference, Seattle, WA, November 1998.

[META96] Proceedings of the Ist IEEE Metadata Conference, Silver Spring, MD, April 1996 (Originally published on the web, Editor: R. Musick, Lawrence Livermore National Laboratory).

[MICH92] Michalewicz, Z., “Genetic Algorithms + Data Structures = Evolutions Programs.”, NY: Springer-Verlag, 1992.

[MINK88] Minker, J., (Editor) "Foundations of Deductive Databases and Logic Programming," Morgan Kaufmann, CA, 1988 (Editor).

[MIT] Technical Reports on Data Quality, Sloan School, Massachusetts Institute of Technology, Cambridge, MA.

[MIT96] Mitchell, M., “An Introduction to Genetic Algorithms.” Cambridge, MA:MIT Press, 1996

[MITC97] Mitchell, T., "Machine Learning," McGraw Hill, NY, 1997.

[MORE98a] Morey, D., "Knowledge Management Architecture," Handbook of Data Management, Auerbach Publications, NY, 1998 (Ed: B. Thuraisingham).

[MORE98b] Morey, D., "Web Mining," Private Communication, Bedford, MA, June 1998.

[MORG88] Morgenstern, M., "Security and Inference in Multilevel Database and Knowledge Base Systems," Proceedings of the 1987 ACM SIGMOD Conference, San Francisco, CA, June 1987.

[NG97] Ng, R., "Image Mining," Private Communication, Vancouver, British Columbia, December 1997.

[NISS96] Panel on Data Warehousing, Data Mining, and Security, Proceedings of the 1996 National Information Systems Security Conference, Baltimore, MD, October 1996.

[NISS97] Papers on Internet Security, Proceedings of the 1997 National Information Systems Conference, Baltimore, MD, October 1997.

[NSF90] Proceedings of the Database Systems Workshop, Report published by the National Science Foundation, 1990 (also in ACM SIGMOD Record, December 1990).

[NSF95] Proceedings of the Database Systems Workshop, Report published by the National Science Foundation, 1995 (also in ACM SIGMOD Record, March 1996).

[NWOS96] Nwosu, K. et al., (Editors) "Multimedia Database Systems, Design and Implementation Strategies." Kluwer Publications, MA, 1996.

[ODMG93] "Object Database Standard: ODMB 93," Object Database Management Group, Morgan Kaufmann, CA, 1993.

[OMG95] "Common Object Request Broker Architecture and Specification," OMG Publications, John Wiley, NY, 1995.

[ORFA94] Orfali, R. et al., "Essential, Client Server Survival Guide," John Wiley, NY, 1994.

[ORFA96] Orfali, R. et al., "The Essential, Distributed Objects Survival Guide," John Wiley, NY, 1994.

[PAKDD97] Proceedings of the Knowledge Discovery in Databases Conference, Singapore, February 1997.

[PAKDD98] Proceedings of the Second Knowledge Discovery in Databases Conference, Melbourne, Australia, April 1998.

[PAW91] Pawlak, Z. "Rough Sets, Theoretical Aspects of Reasoning about Data." Boston: Kluwer Academic Publishers, 1991

[PRAB97] Prabhakaran, B., "Multimedia Database Systems," Kluwer Publications, MA, 1997.

[QUIN79] Quinlan, J.R.):"Discovering Rules from Large Collections of Examples", In *Expert Systems in the Microelectronic Age*, Michie, D. (Ed.), Edimburgo University Press, Edimburgo. 1979

[QUIN86] Quinlan, J.R.: "Induction of Decision Trees (ID3 algorithm)", *Machine Learning J*., vol. 1, núm. 1, pp. 81-106. 1986

[QUIN87] Quinlan, J.R.: "Simplifying decision trees", *International Journal of Man-Machine Studies*, núm. 27, pp. 221-234. 1987

[QUIN88] Quinlan, J.R.: "Decision trees and multivalued attributes", *Machine Intelligence*, núm. 11, pp. 305-318. 1988

[QUIN89] Quinlan, J.R.: "Unknown attribute values in induction". In *Proc. 6th Int. Workshop on Machine Intelligence*, pp. 164-168, Ithaca, NY, June. 1989

[QUIN90] Quinlan, J.R., "Learning logic definitions from relations." *Machine Learning*, 5:139-166, 1990

[QUIN93] Quinlan, J.R., "C4.5: Programs for Machine Learnirig," Morgan Kaufmann, CA, 1993.

[QUIN96] Quinlan, J.R., "Bagging boosting, and C4.5" In Proc. 13 th Natl. Conf Artificial Intelligence (AAAI'96) pages 725-730, Portland, OR, Aug. 1996

[RAMA94] Ramakrishnan, R., (Editor) Applications of Deductive Databases, Kluwer Publications, MA, 1994.

[ROSE98] Rosenthal, A., "Multi-Tier Architecture," Private Communication, Bedford, MA, August 1998.

[RUME86] Rumelhart, D.E., HINTON, G.E. and Williams, R.J., "Learning Internal representations by error propagation." In D.E. Rumelhart and J.L. MacClelland, editors, "Parallel Distributed Processing." Cambridge, Ma: MIT Press 1986

[SIGM96] Proceedings of the ACM SIGMOD Workshop on Data Mining, Montreal, Canada, May 1996.

[SIGM98] Proceedings of the 1998 ACM SIGMOD Conference, Seattle, WA, June 1998.

[SIMO95] Simoudis, E. et al., "Recon Data Mining System," Technical Report, Lockheed Martin Corporation, 1995.

[SQL3] "SQL3," American National Standards Institute, Draft, 1992 (a version also presented by J. Melton at the Department of Navy's DISWG NGCR meeting, Salt Lake City, UT, November 1994).

[STAN98] Stanford Database Group Workshop, Jungalee Virtual Relational Database, September 1998 (also appeared in IEEE Data Engineering Bulletin, June 1998).

[THUR87] Thuraisingham, B., "Security Checking in Relational Database Systems Augmented by an Inference Engine," Computers and Security, Vol. 6, 1987.

[THUR90a] Thuraisingham, B., "Nonmonotonic Typed Multilevel Logic for Multilevel Secure Database Systems," MITRE Report, June 1990 (also published in the Proceedings of the 1992 Computer Security Foundations Workshop, Franconia, NH, June 1991).

[THUR90b] Thuraisingham, B., "Recursion Theoretic Properties of the Inference Problem," MITRE Report, June 1990 (also presented at the 1990 Computer Security Foundations Workshop, Franconia, NH, June 1990).

[THUR90c] Thuraisingham, B., "Novel Approaches to Handle the Inference Problem," Proceedings of the 1990 RADC Workshop in Database Security, Castile, NY, June 1990.

[THUR91] Thuraisingham, B., "On the Use of Conceptual Structures to Handle the Inference Problem," Proceedings of the 1991 IFIP Database Security Conference, Shepherdstown, `WVA, November 1991.

[THUR93] Thuraisingham, B. et al., "Design and Implementation of a Database Inference Controller," Data and Knowledge Engineering Journal, North Holland, Vol. 8, December 1993.

[THUR95] Thuraisingham, B. and Ford, W., "Security Constraint Processing in a Multilevel Secure Distributed Database Management System," IEEE Transactions on Knowledge and Data Engineering, Vol. 7, 1995.

[THUR96a] Thuraisingham, B., "Data Warehousing, Data Mining, and Security (Version I)," Proceedings of the 10th IFIP Database Security Conference, Como, Italy, 1996.

[THUR96b] Thuraisingham, B., "Internet Database Management," Database Management, Auerbach Publications, NY, 1996.

[THUR96c] Thuraisingham, B., "Interactive Data Mining and the World Wide Web," Proceedings of Compugraphics Conference, Paris, France, December 1996.

[THUR97] Thuraisingham, B., " Data Management Systems Evolution and Interoperation," CRC Press, FL, May 1997.

[THUR98] Thuraisingham, B., "Data Warehousing, Data Mining, and Security (Version 2)," Keynote Address at Second Pacific Asia Conference on Data Mining, Melbourne, Australia, April 1998.

[THUR99] Thuraisingham, B., “Data Mining: Technologies, Techniques, Tools and Trends.” CRC Press, 1999

[TKDE93] Special Issue on Data Mining, IEEE Transactions on Knowledge and Data Engineering, December 1993.

[TKDE96] Special Issue on Data Mining, IEEE Transactions on Knowledge and Data Engineering, December 1996.

[TRUE89] Trueblood, R. and Potter, W., "Hyper-Semantic Data Modeling," Data and Knowledge Engineering Journal, Vol. 4, North Holland, 1989.

[TSUR98] Tsur, D. et al., "Query Flocks: A Generalization of Association Rule Mining," Proceedings of the 1998 ACM SIGMOD Conference, Seattle, WA, June 1998.

[TSIC82] Tsichritzis, D. and Lochovsky, F., "Data Models," Prentice Hall, NJ, 1982.

[ULLM88] Ullman, J. D., "Principles of Database and Knowledge Base Management Systems," Volumes I and 11, Computer Science Press, MD 1988.

[UTG88] Utgoff, P.E., "An Incremental ID3." In Proc. Fifth Int. Conf. Machine Learning, pages 107-120, San Mateo, CA, 1988

[VIS95] Proceedings of the 1995 Workshop on Visualization and Databases, Atlanta, GA, October 1997 (Ed: G. Grinstein)

[VIS97] Proceedings of the 1997 Workshop on Visualization and Data Mining, Phoenix, AZ, October 1997 (Ed: G. Grinstein).

[VLDB98] Proceedings of the Very Large Database Conference, New York City, NY, August 1998.

[WIED92] Wiederhold, G., "Mediators in the Architecture of Future Information Systems," IEEE Computer, March 1992.

[WOEL86] Woelk, D. et al., "An Object-Oriented Approach to Multimedia Databases," Proceedings of the ACM SIGMOD Conference, Washington DC, June 1986.

[XML98] Extended Markup Language, Document published by the World Wide Web Consortium, Cambridge, MA, February 1998.